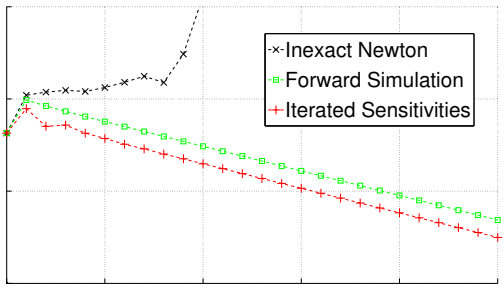
KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Numerical Simulation Methods for Embedded Optimization



Rien Quirynen

Dissertation presented in partial fulfillment of the requirements for the joint degree of Doctor in Engineering Science (KU Leuven) and Doctor Rerum Naturalium (Freiburg)

January 2017

Numerical Simulation Methods for Embedded Optimization

Rien QUIRYNEN

Examination committee:
Prof. Dr. A. Bultheel, chair
Prof. Dr. M. Diehl, supervisor
(University of Freiburg)
Prof. Dr. S. Vandewalle, supervisor
Prof. Dr. J. Suykens
Ass. Prof. Dr. P. Patrinos
Ass. Prof. Dr. G. Pipeleers
(all KU Leuven)
Prof. Dr. S. Bartels
Prof. Dr. D. Kröner
(all University of Freiburg)
Prof. Dr. Dres. h.c. H. G. Bock
(Heidelberg University)

Dissertation presented in partial fulfillment of the requirements for the joint degree of Doctor in Engineering Science and Doctor Rerum Naturalium

Arenberg Doctoral School
Faculty of Engineering Science
KU Leuven

Faculty of Mathematics and Physics
University of Freiburg

January 2017

Acknowledgements

As I believe it should mostly be the case, it is with strongly mixed feelings that I write this final section of my PhD thesis. On the one hand, of course, relief and excitement about what the future still holds. While on the other hand, nostalgia and also astonishment about how fast this period has gone by. I can safely say that the last four years have formed a challenging but mostly exciting journey that allowed me to broaden my horizon on both a personal and professional level. I learned so many new things, experienced new places and this while meeting the most amazing people. For this, I have to thank a lot of people that have contributed to this experience.

My first and greatest thanks goes to Moritz Diehl, without whom you would most likely not have been reading my PhD thesis to begin with. He provided me with the initial spark of inspiration and motivation to perform research, starting with my final master project under his supervision. Thanks to his clear passion for research and the very flexible and comfortable working environment that he creates, I discovered my own passion and determination to pursue a PhD. Throughout this experience, Moritz provided me with many more research ideas than one could possibly explore in one PhD project, with the support and flexibility to direct my own research path and the continuous source of enthusiasm and inspiration which makes him such an exceptional supervisor. I also thank Moritz for creating a very fruitful environment for cooperation and exchange of research ideas within the group, which greatly profits from a particular attention to creating an enjoyable working atmosphere.

I owe my sincere gratitude to Stefan Vandewalle for his valuable feedback and guidance, and especially for agreeing to become my supervisor at KU Leuven. In addition to my two main supervisors, I would also like to thank Sébastien Gros and Boris Houska, who contributed strongly by mentoring me both on a personal and professional level. Sébastien was a postdoctoral researcher in our group when I started and he helped me with becoming a productive researcher from the very start of my PhD project. But also after he eventually became

assistant professor in Chalmers, we stayed in contact and continued a fruitful cooperation. Similarly, I met Boris as a very bright and inspiring PhD researcher in the same group when I was still a master student myself. I feel privileged to have worked together on a daily basis during my research time in Shanghai and also afterwards, I could always count on his quick but insightful feedback and contributions to joint publications.

I would like to thank my colleagues and fellow researchers both in Leuven and Freiburg but also during my visit in Geneva and Shanghai, for the inspiring discussions and collaborations, the relaxing (coffee) breaks and generally the great times we had together in- and outside of the office. From my research period in Leuven, I especially want to thank Adeleh, Andrew, Attila, Boris, Greg, Janick, Joachim, Joel, Joris, Kurt, Mario, Milan, Quoc, Sébastien and Slava. I would like to thank Adrian, Andrea, Dang, Dimitris, Fabian, Gianluca, Gianni, Jochem, Jonas, Jörg, Michael, Mikhail, Rachel, Robin and Thor for creating a fruitful and enjoyable working environment together in Freiburg. Considering the multiple locations where I gained a network of friends and colleagues, it is unlikely that the above lists are close to complete.

In addition, I would like to thank both Thiva Albin and Dennis Ritter from the control research group of Dirk Abel at the Aachen University, for our fruitful collaboration resulting in multiple exciting publications. Especially, I would like to express my gratitude to agree on using our algorithms and software for their challenging real-world problem and on their experimental test setup, with the corresponding results in this thesis. I am indebted to Dimitris, Fabian, Rachel, Robin and Thor, as well as both of my supervisors for providing very valuable comments and remarks on earlier versions of this manuscript. I would also like to thank all the members of my PhD jury for reading my thesis and for providing crucial feedback on my research results.

I gratefully acknowledge funding by the Research Foundation – Flanders (FWO), through a personal 2×2 years PhD fellowship. I would additionally like to thank all people involved at the FWO for their very smooth organization of all administrative tasks, especially considering the multiple locations where I have stayed during my PhD project. Also the funding by the DFG, via the project “Numerical methods for optimization based control of cyclic processes”, as part of the research unit FOR 2401 during the last three months of the PhD is gratefully acknowledged. I would like to thank Leen Cuypers, Jacqueline De bruyn, Elsy Vermoesen and John Vos from KU Leuven and Daniela Högerle, Christine Paasch and Kerstin Pfeiffer from the University of Freiburg for their crucial support with all administrative tasks and for making this joint PhD possible. I also want to thank all people involved in the EMBOCON, SADCO, HIGHWIND, TEMPO and AWESCO projects, since I was able to attend and profit from many of their events with my own funding from FWO. Through

these various research networks, I met a lot of interesting people with whom I had inspiring interactions and we had great times together.

In addition, I gratefully acknowledge the funding from the Junior Mobility Programme (JuMo) in the form of a YouReCa travel grant at the KU Leuven. Within this project on the topic of "Integration Methods for Real-Time Multiple Shooting", I was able to spend about four months at the University of Geneva and three more months at the Shanghai Jiao Tong University in the period of 2013-2014. I especially thank Martin Gander and Boris Houska for hosting me respectively in Geneva and Shanghai and for providing me with the chances to interact and have interesting discussions with many other researchers. I would additionally like to thank all the amazing people that I met during my time at both locations, in- and outside of the university, since they made my stays such an enjoyable and memorable experience.

Last but surely not least, I would like to thank all my friends on whom I could count for support but even more for pleasant distractions from any workaholic tendencies. Especially in the final stage of writing my thesis, many outdoor activities around Freiburg reminded me of the beauty and adventures to be had outside of LaTeX. I most definitely cannot imagine this continuing journey without the endless support from my amazing parents, Marleen and Willy, as well as from both of my sisters, Geertrui and Melle.

Rien Quirynen

Abstract

Dynamic optimization based control and estimation techniques have gained increasing popularity, because of their ability to treat a wide range of problems and applications. They rely on the explicit formulation of a cost function, which needs to be minimized given the constraints of the problem and the system dynamics. Especially in the context of real-time applications of control and estimation on embedded hardware, the computational burden associated with the online solution of the optimal control problem forms the main limiting factor in the deployment of such an advanced strategy.

For that purpose, this thesis considers the development of tailored algorithms of simulation methods for embedded optimization that allows for an efficient implementation of nonlinear model predictive control (NMPC) or moving horizon estimation (MHE). A direct treatment of the optimal control problem requires the numerical simulation of the continuous time nonlinear dynamics and the solution of the resulting large but structured optimization problem. We additionally propose a new format to define the dynamic model, which allows one to directly exploit the present structure of linear or partially linear subsystems in the formulation of the system dynamics. In addition, we also discuss embedded optimization algorithms for a more general set of interconnected subsystems based on a distributed multiple shooting technique.

As we focus on Newton-type optimization algorithms, it is important to extend the numerical simulation method with an efficient propagation of its first and possibly higher order derivative information. We discuss the tailored implementation of such a sensitivity analysis for both explicit and implicit integration, such as the collocation methods that form a specific family of implicit Runge-Kutta schemes. In addition, a novel Hessian propagation scheme is proposed for both a discrete and continuous time sensitivity analysis, which allows one to maintain and exploit the symmetry of the second order derivatives. Based on these symmetric sensitivity equations, an alternative three-sweep propagation (TSP) technique is presented and analyzed.

When embedding an implicit integration scheme within a direct multiple shooting based Newton-type optimization algorithm, one ends up with an outer and inner level of iterations which is typically not the most efficient computational approach. We therefore propose an alternative implementation, which we refer to as a lifted collocation integrator, and discuss its advantages and disadvantages compared to multiple shooting and direct collocation. Two alternative extensions to inexact Newton based optimization are presented, using either an adjoint differentiation technique or an iterative sensitivity propagation. We establish new theoretical results on the local convergence of this inexact Newton scheme with iterated sensitivities. Unlike for previously existing algorithms, we show that local convergence for the inner scheme is necessary and often also sufficient for asymptotic contraction of the new proposed optimization method.

In addition to the use of tailored optimal control algorithms, the performance of embedded applications strongly relies on efficient code implementations. This thesis therefore includes an implementation of the major new algorithmic techniques as part of the automatic code generation tool within the open-source `ACADO Toolkit` software package. We discuss some of the main real-world control applications that were made possible using such an `ACADO` code generated solver. More specifically, we discuss the airpath control for a two-stage turbocharged gasoline engine in more detail. The resulting NMPC scheme on the dSpace MicroAutoBox is shown to meet the challenging demands of this control application, with a sampling time of 25 ms. It is validated based on closed-loop simulations as well as in-vehicle experimental results.

Beknopte Samenvatting

Dynamische optimalisatie gebaseerde controle- en schattingstechnieken worden steeds meer populair, omwille van hun vermogenm een wijde selectie aan problemen en toepassingen te behandelen. Ze zijn afhankelijk van de expliciete formulering van een kostfunctie, welke moet worden geminimaliseerd onder de beperkingen van het probleem en de bijhorende systeemdynamica. Vooral in de context van real-time toepassingen van controle en schatting op ingebedde hardware, vormt de rekenkundige last voor het online oplossen van het optimaal controleprobleem een belangrijke beperkende factor in de implementatie van een dergelijke geavanceerde strategie.

Deze thesis beschouwt daarom de ontwikkeling van op maat gemaakte algoritmen voor ingebedde optimalisatie- en simulatiemethoden, welke een efficiënte implementatie toelaten van niet-lineaire model predictieve controle (NMPC) of bewegende horizon schatting (MHE). Een directe behandeling van het optimaal controleprobleem vereist de numerieke simulatie van de niet-lineaire dynamica in continue tijd en de oplossing van het resulterend, groot maar gestructureerd optimalisatieprobleem. We stellen bovendien een nieuwe formulering van het dynamisch model voor, om de aanwezige structuur van (gedeeltelijk) lineaire deelsystemen onmiddellijk te kunnen herkennen en dan ook te benutten. Bovendien bespreken we ook het gebruik van ingebedde optimalisatie algoritmen voor een meer algemene reeks van onderling verbonden deelsystemen, en dit op basis van een gedistribueerde variant van multiple shooting.

Gezien we ons richten op Newton gebaseerde optimalisatie algoritmen, is het belangrijk om de numerieke simulatiemethode uit te breiden met een efficiënte propagatie van de bijhorende eerste en hogere orde afgeleiden. We bespreken daarom de aangepaste implementatie van een dergelijke sensitiviteitsanalyse voor zowel expliciete als impliciete integratie, zoals bijvoorbeeld de collocatiemethoden die een specifieke klasse vormen van impliciete Runge-Kutta schema's. Daarnaast presenteren we ook een nieuw schema voor de specifieke propagatie van de Hessiaan en dit zowel in discrete

als in continue tijd. Het laat ons toe om de symmetrie van deze tweede orde afgeleiden te behouden en dan ook te benutten. Op basis van deze symmetrische vergelijkingen, kan een alternatieve three-sweep propagatie (TSP) techniek worden gebruikt voor de sensitiviteitsanalyse.

Bij het gebruik van een impliciet integratieschema binnen een Newton gebaseerd optimalisatie algoritme, eindigt men met een buitenste en binnenste niveau van iteraties en dat is typisch niet de meest efficiënte berekeningsaanpak. We stellen daarom een alternatieve implementatie voor, welke we lifted collocatie noemen, en we discussiëren de voor- en nadelen in vergelijking met multiple shooting en directe collocatie. Twee alternatieve uitbreidingen zijn mogelijk voor inexacte Newton gebaseerde optimalisatie, door gebruik van ofwel een achterwaartse differentiatie techniek ofwel een iteratieve sensitiviteitsanalyse. We presenteren nieuwe theoretische resultaten in verband met de lokale convergentie van dit inexact Newton schema met iteratieve sensitiviteiten. In tegenstelling tot standaard algoritmen, kunnen we aantonen dat de lokale convergentie voor het binnenste schema nodig is en vaak ook voldoende voor de asymptotische contractie van de voorgestelde optimalisatiemethode.

Naast het gebruik van op maat gemaakte optimale controlemethoden, is de numerieke prestatie voor ingebedde toepassingen sterk afhankelijk van efficiënte implementaties. Deze thesis bevat daarom ook een software implementatie van de nieuwe algoritmische technieken als onderdeel van de automatische code generatie tool binnen de open-source `ACADO Toolkit`. We vermelden een aantal van de belangrijkste controle toepassingen, die mogelijk werden gemaakt dankzij `ACADO` gebaseerde code. Meer specifiek, bespreken we bijvoorbeeld de airpath besturing voor een tweefasige turbo benzinemotor. Het resulterend NMPC algoritme op de dSpace MicroAutoBox kan de uitdagende eisen van deze controle toepassing verwezenlijken, zoals bevestigd op basis van closed-loop simulaties en experimentele resultaten met een echt voertuig.

Zusammenfassung

Auf numerischer Optimierung beruhende Regelungs- und Schätzungsverfahren erfreuen sich zunehmender Beliebtheit aufgrund der erfolgreichen Verwendung in zahlreichen Anwendungen. Diese Verfahren basieren auf einer explizit formulierten Kostenfunktion, die unter Einhaltung von problemspezifischen Randbedingungen und der dynamischen Eigenschaften des Systems minimiert wird. Besonders bei Echtzeit-kritischen Anwendungen auf eingebetteten Systemen ist der erhöhte Rechenaufwand, verursacht durch die erforderliche Lösung eines optimalen Steuerungsproblems, ein limitierender Faktor für die Verwendung einer solch fortgeschrittenen Regelungsstrategie.

Aus diesem Grund handelt diese Arbeit von der Entwicklung maßgeschneiderter Algorithmen für Optimierungs- und Simulationsmethoden für eingebettete Systeme, die eine effiziente Implementierung von Algorithmen aus der nichtlinearen modellbasierten prädiktiven Regelung (NMPC) und Zustandsschätzung auf bewegten Horizonten (MHE) ermöglichen. Diese Verfahren erfordern die Simulation der zeitkontinuierlichen und häufig nichtlinearen Systemdynamik sowie die Lösung eines großen, jedoch strukturieren Optimierungsproblems. Hierfür wird eine neue Formulierung des dynamischen Modells eingeführt, die es erlaubt lineare oder teilweise lineare Komponenten zu erkennen und effizient zu nutzen. Basierend auf der Theorie des verteilten Mehrschussverfahrens werden darüber hinaus Optimierungsalgorithmen speziell für Netzwerke aus eingebetteten Systemen vorgestellt.

Für die hier betrachteten Newton-basierenden Optimierungsalgorithmen ist eine Erweiterung der Simulationsmethoden um das effizientes Propagieren von Ableitungen erster und eventuell höherer Ordnung von großer Bedeutung. Maßgeschneiderte Implementierungen für eine Sensitivitätsanalyse von expliziten und impliziten Integrationsverfahren, sowie von Kollokation Verfahren werden diskutiert. Letztere bilden eine spezielle Gruppe von impliziten Runge-Kutta Verfahren. Zudem wird eine neue Methode für die Propagation der Hesse Matrix vorgestellt, die sowohl eine diskrete als auch kontinuierliche Sensitivitätsanalyse

erlaubt und dabei vorhandene Symmetrien in der zweiten Ableitung ausnutzt und erhält. Basierend auf diesen symmetrischen Sensitivitäts-Gleichungen wird ein alternativer, auf drei Durchläufen basierender Ansatz (TSP) analysiert.

Durch die Verwendung eines impliziten Integrationsverfahrens innerhalb eines Newton-basierten Optimierungsalgorithmus entstehen Iterationen auf zwei verschiedenen Stufen. Das Lösen des Problems auf der inneren sowie der äußeren Stufe bringt einen erhöhten Rechenaufwand mit sich. In dieser Arbeit wird eine alternative Implementierung namens geliftete Kollokations-Integratoren vorgestellt und die Vor- und Nachteile des Verfahrens werden im Vergleich zur Mehrschuss- und direkte Kollokation Methode erläutert. Zwei alternative Erweiterungen zur inexakten Newton-basierten Optimierung werden eingeführt, wobei eine die Methodik der Rückwärts-Differenzierung und die andere eine iterativen Propagation der Sensitivitäten verwendet. Das iterative Verfahren wird um einen theoretischen Beweis für die lokale Konvergenz ergänzt. Im Gegensatz zu bereits existierenden Algorithmen wird bewiesen, dass lokale Konvergenz des inneren Problems notwendig, und häufig auch hinreichend ist, um eine asymptotische Kontraktion des vorgestellten Newton-basierten Optimierungsverfahrens zu erzielen.

Zusätzlich zu angepassten Algorithmen für die optimierungsbasierte Regelung ist eine effiziente Implementierung notwendig, um die erforderliche Performanz auf eingebetteten Systemen zu garantieren. Aus diesem Grund umfasst diese Arbeit, die Integration der bedeutenden algorithmischen Erkenntnisse in das quelloffene `ACADO Toolkit`. Die Software generiert problemspezifischen Quellcode, der wiederum verwendet werden kann um gegenwärtige Regelungsprobleme zu lösen. Im Detail wird die Regelung des Luftstroms eines zweistufig aufgeladenen Turbomotors diskutiert. Es wird gezeigt, dass das hieraus resultierende NMPC Regelungssystem auf einer dSpace MicroAutoBox den hohen Anforderungen dieser Anwendung gerecht wird, was in Simulationen und in Fahrzeug-Experimenten validiert wird.

Abbreviations

BLAS	Basic Linear Algebra Subprograms
CPU	central processing unit
FLOP	floating point operation
FPGA	field-programmable gate array
LGPL	GNU Lesser General Public License
iff	if and only if

Dynamic Systems

DAE	Differential-Algebraic Equations
IVP	Initial Value Problem
LTV	Linear Time Varying
MIMO	multiple-input multiple-output
NARX	Nonlinear AutoRegressive eXogenous model
ODE	Ordinary Differential Equations
PDE	Partial Differential Equations
PID	proportional–integral–derivative
SISO	single-input single-output

Numerical Simulation

AB	Adams-Bashforth
AM	Adams-Moulton
BDF	Backward Differentiation Formula
DIRK	Diagonally Implicit RK
DOPRI	Dormand–Prince
ERK	Explicit Runge-Kutta
ESDIRK	Explicit SDIRK
IRK	Implicit Runge-Kutta
LM	Linear Multistep
MOL	Method Of Lines
RKF	Runge–Kutta–Fehlberg

RK	Runge–Kutta
SDIRK	Singly Diagonally IRK

Sensitivity Analysis

END	External Numerical Differentiation
FB	Forward-Backward propagation
FOA	Forward-Over-Adjoint
IFT	Implicit Function Theorem
IND	Internal Numerical Differentiation
TSP	Three-Sweep Propagation
VDAE	Variational Differential-Algebraic Equations
VDE	Variational Differential Equations

Optimization Theory

FONC	First Order Necessary Conditions
KKT	Karush-Kuhn-Tucker
LICQ	Linear Independence Constraint Qualification
NLP	Nonlinear Programming
QP	Quadratic Programming
SOSC	Second Order Sufficient Conditions

Newton-Type Optimization

AF-INIS	Adjoint-free INIS
BFGS	Broyden-Fletcher-Goldfarb-Shanno
GGN	Generalized Gauss-Newton
INIS	Inexact Newton with Iterated Sensitivities
IN	Inexact Newton
IP	Interior Point
SCP	Sequential Convex Programming
SQP	Sequential Quadratic Programming

Optimal Control

BVP	Boundary Value Problem
CMS	Centralized Multiple Shooting
DMS	Distributed Multiple Shooting
DP	Dynamic Programming
HJB	Hamilton-Jacobi-Bellman
MHE	Moving Horizon Estimation
MPC	Model Predictive Control
MS	Multiple Shooting
OCP	Optimal Control Problem
RTI	Real-Time Iteration

List of Symbols

Direct Optimal Control

z_k	discrete-time algebraic variables
$l(\cdot)$	discrete-time stage cost
N_s	number of integration steps per interval
T_int	integration step size
T_s	discretization length
x_k	discrete-time differential states
K	collocation variables
N	number of shooting intervals
q	number of collocation nodes

Linear Algebra

$\mathbb{0}$	zero matrix
$\mathbb{1}$	identity matrix
$\mathbb{N}$	set of natural numbers
$\otimes$	Kronecker product
$\mathbb{R}$	set of real numbers
$\rho(\cdot)$	spectral radius
$\mathbb{S}^n$	set of $n \times n$ symmetric matrices
$\mathbb{S}^n_{++}$	set of $n \times n$ positive definite matrices

$\sigma(\cdot)$	spectrum of a matrix
$\|\cdot\|$	norm function

Nonlinear Programming

$c(\cdot)$	equality constraints
$h(\cdot)$	inequality constraints
$\mathcal{F}(\cdot)$	first order necessary conditions
κ	local contraction rate
$\mathcal{L}(\cdot)$	Lagrangian function
$\psi(\cdot)$	objective function

Dynamic Optimization

$\ell(\cdot)$	continuous-time stage cost
$h(\cdot)$	path inequality constraints
$m(\cdot)$	mayer cost term
$r(\cdot)$	terminal constraints
$R(\cdot)$	residual function
T	time horizon length

Dynamic Systems

$f_{\mathrm{e}}(\cdot)$	explicit set of differential equations
$f(\cdot)$	implicit set of differential equations
$g(\cdot)$	implicit set of algebraic equations
$u(t)$	continuous-time control inputs
$x(t)$	continuous-time differential states
$\dot{x}(t)$	continuous-time differential state derivatives
$y(t)$	continuous-time system outputs
$z(t)$	continuous-time algebraic variables

Contents

List of Figures

List of Tables

Introduction

Our quality of life, the world's productivity and its sustainability become more and more determined by the outcome and benefits of process automation. In this domain of automatic control, an important distinction can be made between *offline* and *online* tasks. The offline context refers to the problems that can be solved while the process is not yet running, such as the system design, analysis and identification. These activities are not limited by real-time feasibility constraints and typically require considerable amounts of brainwork and computer-human interaction. It includes the implementation and tuning of an automatic control scheme, which can be used to stabilize the system and, for example, steer it to a specific desired state. The latter is referred to as a *controller* for the dynamic system and executes itself continuously as long as the process is running, i.e., it forms part of the online context.

This work is concerned mostly with the online problems related to the control of fast dynamic systems. In addition to the task of computer based automatic control, this includes online state and system parameter estimation. Since many systems operate in a repetitive manner such as, for example, robot arm manipulators or chemical batch processes, the field of iterative learning can play an important role. In this context, the control performance can be improved iteratively based on previous repetitions. In addition to this online information that can be either learned, measured or estimated, there are typically many remaining effects or disturbances which cannot be taken into account. This is the motivation for closing the loop with *feedback control*, in order to reject such unknown disturbances or any mismatch. A typical case is when the difference between the observed system output and the desired reference is provided as the feedback signal to the controller.

A popular viewpoint among researchers on mathematical optimization is that all relevant decisions can and should be cast into an optimization problem. Based on a quantitative representation and suitable approximation of the considered environment, one can typically identify a certain objective or performance index

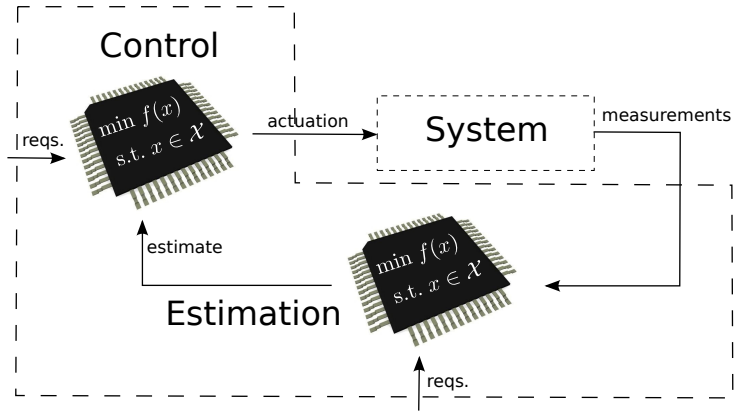


Figure 1: Illustration of the considered principle of using embedded optimization for real-time estimation and feedback control.

and corresponding constraints or limitations of the problem task. Many of these cases result in static optimization formulations, i.e., without explicitly considering dynamic system evolutions. Application examples of the latter include supply chain management, scheduling or portfolio optimization in economics as well as setpoint calculation, optimal design and regression analysis in modeling and engineering. On the other hand, throughout this thesis, we are more interested in dynamic optimization problems which directly take a model for the dynamic process behaviour into account. In the context of systems engineering, the resulting field is typically referred to as *optimal control*. The corresponding formulation of a dynamic optimization problem consists of the system dynamics in addition to the problem objective and constraint functions. Examples of optimal control applications include the optimization of an actuation profile in order to perform a certain task, system state and parameter estimation or optimal experiment design.

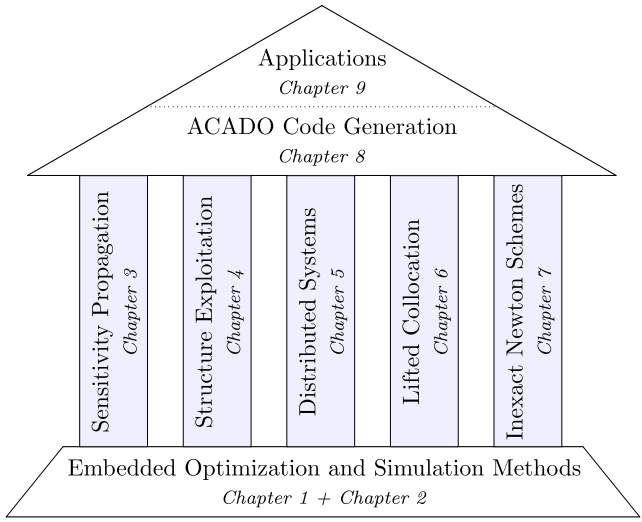
This thesis focuses more specifically on the real-time optimal control of nonlinear small to medium-scale systems with relatively fast dynamics, in order to design an advanced feedback control strategy. For this purpose, both the online estimation and control task can be carried out by embedded dynamic optimization tools. Technically, the term *embedded* in this case refers to the situation of having any dedicated computer system as part of a larger (typically mechanical or electrical) device, often with limited processing resources and with real-time computing constraints. Figure 1 illustrates the resulting closed-loop system. In the context of model predictive control (MPC) and moving horizon estimation (MHE), both the control and estimation task correspond to

an online sequence of dynamic optimization problems based on the receding horizon principle. More specifically, the system parameters or actuation profile are optimized over a certain time horizon using specific measurements and directly taking into account the problem objective, the system dynamics and additional constraints or specifications. These desirable properties also account for the increasing popularity of such dynamic optimization based closed-loop control and estimation techniques.

The main disadvantage of embedded optimization based approaches is the resulting computational effort, especially in case of complex nonlinear problems. One needs to solve a large, typically non-convex dynamic optimization problem online and this under strict timing constraints in order to stabilize and control the potentially fast process dynamics. One attractive alternative therefore relies on the explicit and offline computation of the resulting feedback control law, even though this approach unfortunately suffers from the curse of dimensionality when solving this problem and storing the results. Instead, we focus on the direct and online solution of the optimal control problem at each sampling time point. In order to widen the range of applications that can be treated by using such online optimization based techniques, it becomes crucial to use efficient implementations of tailored algorithms on suitable computational hardware. *For that reason, this thesis considers the development of real-time feasible numerical algorithms for embedded optimization and it includes open-source software implementations.* In the case of complex nonlinear systems with fast dynamics, the numerical simulation of the continuous time differential equations forms an important computational component of any direct optimal control method. In addition, the use of derivative based optimization algorithms results in the need for an efficient propagation of corresponding first and possibly higher order sensitivity information.

Contributions and Outline of this Thesis

Let us introduce the structure of this thesis, by describing the specific topics that are discussed in the different chapters as well as their corresponding main contributions.



Chapter 1 - Fast Nonlinear Model Predictive Control and Estimation This chapter introduces the class of dynamic systems and the corresponding optimal control problem formulation, which we handle throughout this thesis. It provides a detailed overview on direct optimal control, including a discussion on the theory and numerical algorithms for nonlinear programming. In addition, the current state of the art is presented regarding tailored solvers for optimal control and online algorithms for fast nonlinear MPC and MHE. More specifically, the Real-Time Iteration (RTI) scheme is introduced which forms the algorithmic framework for many of the new developments in the following chapters.

Chapter 2 - Numerical Simulation and Sensitivity Propagation In direct optimal control, one relies on a numerical integration scheme to simulate the system of differential equations. This chapter provides an overview on numerical simulation methods with a specific focus on explicit and implicit Runge-Kutta formulas. Within a Newton-type optimization method, it also becomes crucial to accurately and efficiently evaluate first and possibly higher order sensitivities for each simulation result. Different approaches for performing the corresponding sensitivity analysis are presented and compared for both explicit and implicit integration schemes. A detailed discussion is provided on collocation methods and their efficient implementation for embedded optimization. In addition to the introduction of these concepts, the chapter contributes new optimal control applications based on continuous output. For example, it shows that the continuous output feature allows for an efficient implementation of closed-loop costing for MPC and of an MHE scheme with multi-rate measurements.

The first two chapters introduced and provided an overview of the state of the art regarding numerical algorithms for both embedded optimization and simulation. Building on these concepts, all subsequent chapters present the main contributions of the thesis.

Chapter 3 - Symmetric Hessian Propagation Technique This chapter presents an efficient second order differentiation scheme for both discrete- and continuous-time sensitivity analysis. A novel Hessian propagation technique is proposed, which allows one to maintain and exploit the symmetry of the directional derivative contributions for any explicit or implicit integration method. In addition, the discussion in a continuous-time framework allows for a generic sensitivity analysis before applying a numerical discretization scheme. Based on the proposed symmetric sensitivity equations, a new alternative three-sweep Hessian propagation (TSP) scheme will be presented. Unlike the classical forward-backward approach, this technique can result in a considerable reduction of the corresponding memory requirements. An implementation of these symmetric Hessian propagation techniques in the open-source `ACADO Toolkit` software is presented and its performance is illustrated on the case study of a nonlinear biochemical reactor.

Chapter 4 - Structure Exploitation for Linear Subsystems The chapter proposes a new format for defining nonlinear dynamic models and shows how the three-stage structure therein can be strongly exploited especially by implicit integration methods. The structure of linear subsystems in the differential equations is introduced and motivated, followed by a discussion on the consequences for the implementation of direct optimal control methods. Two real-world control examples are used to illustrate the relevance of the proposed three-stage model structure. Numerical experiments show that considerable speedups can be achieved with these structure exploiting integrators, which are implemented as part of the `ACADO` code generation tool.

Chapter 5 - Compression Algorithm for Distributed Multiple Shooting We propose and discuss algorithmic techniques for the efficient implementation of real-time feasible NMPC for decomposable systems based on Distributed Multiple Shooting (DMS). The chapter describes how tailored collocation based integrators with continuous output can be used to efficiently parameterize the coupling between subsystems. In addition, a novel compression algorithm is presented as an effective way of reducing the computational burden for the embedded convex solver. As a particular example, we discuss the numerical exploitation of a tridiagonal coupling structure. The resulting DMS-RTI

scheme is implemented based on the ACADO code generation and is illustrated using the example of a chain of spring connected masses. Already within a serial implementation, the presented algorithm is shown to be able to provide impressive speedups over the conventional scheme.

Chapter 6 - Lifted Newton-Type Collocation Integrators We present an implicit extension of the lifted Newton method for collocation schemes and discuss its connection to multiple shooting and direct collocation in a general framework, independent of the Newton-type optimization method. It allows us to interpret this technique as a direct and parallel exploitation of the specific collocation problem structure, for which we discuss both the advantages and disadvantages over classical approaches. In addition, the chapter proposes and analyzes two alternative approaches for inexact Newton based lifted collocation, using either an adjoint derivative propagation or iterated forward sensitivities. A more practical contribution of this chapter is the open-source implementation of these novel lifting schemes within the ACADO code generation tool for embedded optimal control applications. The numerical performance of the proposed algorithmic techniques is illustrated on the benchmark case study of the optimal control for a chain of masses.

Chapter 7 - Local Convergence of Inexact Newton with Iterated Sensitivities The chapter presents the Inexact Newton method with Iterated Sensitivities (INIS) in the most general nonlinear programming framework in order to discuss its local convergence. We show that this scheme allows us to recover a strong connection between the local contraction rate of the forward problem and the local convergence properties of the resulting Newton-type optimization algorithm. Unlike the case for standard inexact Newton methods, local contraction based on the Jacobian approximation for the forward problem is necessary and, under mild conditions, even sufficient for local convergence of the INIS optimization scheme. In addition, an adjoint-free (AF-INIS) variant for Newton-type optimization is proposed and its local convergence properties are also studied. This alternative approach is preferable whenever the algorithm can be carried out independently of the respective values for the multipliers corresponding to the equality constraints, but it generally does not preserve the local convergence properties of the forward scheme. Finally, the numerical performance and theoretical results for the presented algorithms are illustrated based on an optimal control case study.

Chapter 8 - Open-Source ACADO Code Generation Software As mentioned earlier, an important contribution of the thesis consists of the implementation

of the presented new algorithmic techniques as part of the open-source ACADO code generation software package. This chapter therefore introduces the concept of automatic code generation and discusses the implementation as part of the ACADO Toolkit. In addition, we describe the typical workflow needed to export a tailored optimal control solver and we present the ACADO integrator suite, which forms the main software result of this thesis, in more detail. The tool provides the option to generate a stand-alone, embeddable integrator code with tailored sensitivity propagation, which can be used, e.g., to prototype new algorithms. We discuss some of the many real-world control applications that were made possible by using the newly developed algorithmic techniques within the open-source ACADO code generation tool.

Chapter 9 - Two-Stage Turbocharged Gasoline Engine A particularly challenging optimal control application consists of the development of a nonlinear MPC (NMPC) scheme which allows one to meet the tough demands on closed-loop control of a two-stage turbocharged gasoline engine. This forms a promising airpath concept in order to reduce the fuel consumption, even though it exhibits strong nonlinear behaviour while having high demands on the control quality and system limitations. The overall goal is that the control scheme makes use of the specific turbocharging architecture to overcome the trade-off between a fast transient raise of the boost pressure and a high specific power. The main contribution in this chapter is the proposed implementation of a real-time feasible NMPC algorithm, with a sampling time of 25 ms, based on an ACADO code generated solver on the dSpace MicroAutoBox as the embedded control hardware. The resulting system is thoroughly validated by performing closed-loop simulations and real-world experiments using the implementation within a demonstrator vehicle, both on a dynamometer and on the road.

Chapter 1

Fast Nonlinear Model Predictive Control and Estimation

Computation times below one microsecond are typically associated with explicit approaches for linear Model Predictive Control (MPC), for example, as presented in [30] and implemented in the Multi-Parametric Toolbox `MPT3` [163]. Given the modern computer architectures and the algorithmic techniques presented in this chapter, it is however possible to achieve such real-time requirements using online, embedded optimization for relatively small applications. Based on the control example from [68], numerical results for Nonlinear MPC (NMPC) simulations with a sampling time below one microsecond have been presented in [270]. Moreover, the proposed direct methods do not suffer from the curse of dimensionality [29] in the same way as explicit solution strategies, such that also larger applications can be considered [315].

Outline The chapter is organized as follows. Section 1.1 briefly presents the class of systems and corresponding dynamic models in which we are interested. The optimal control problem formulation and direct solution approaches are introduced in Section 1.2, followed by an overview on nonlinear programming and Newton-type optimization in Section 1.3. Sequential quadratic programming can rely on structure exploiting solvers for optimal control, as described in Section 1.4. Finally, Section 1.5 presents online algorithms that allow real-time implementations of fast nonlinear model predictive control and estimation.

1.1 Controlled Dynamic Systems

Let us start by introducing the class of dynamic systems, which we are interested in throughout this thesis. We mostly consider *controlled dynamic systems*, i.e., systems that can be manipulated externally. In most of this thesis, the particular nature of this controlled system will not be specified even though the focus is on fast dynamic systems with sampling times in the milli- or even microsecond range. A typical domain of application therefore consists of mechatronic systems (see Chapter 9) but the proposed algorithmic techniques can also be used for other classes of dynamic systems [218].

The system of interest can be described by a deterministic set of differential equations. Note that our discussion further omits other interesting modeling classes for dynamic systems, such as stochastic differential equations [33] or black box modeling [216, 293]. Instead, we focus on systems of *Ordinary Differential Equations* (ODE) and *Differential-Algebraic Equations* (DAE) which will be introduced in the next two subsections.

1.1.1 Ordinary Differential Equations

The following definition introduces an implicit ODE system.

Definition 1.1 (Ordinary Differential Equations) *We define the time $t \in \mathbb{R}$, the differential states $x(t) \in \mathbb{R}^{n_\mathrm{x}}$ and control inputs $u(t) \in \mathbb{R}^{n_\mathrm{u}}$. A system of Ordinary Differential Equations (ODE) to describe the dynamic evolution of this state vector $x(t)$ then reads as*

$$0 = f(\dot{x}(t), x(t), u(t)), \tag{1.1}$$

where the notation $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$ is used for the differential state derivatives and the Jacobian matrix $\frac{\partial f}{\partial \dot{x}}(\cdot)$ is assumed to be invertible, $\forall t$.

Note that an explicit system of ODE equations belongs to a special subclass, where the state instead is described by

$$\dot{x}(t) = f_\mathrm{e}(x(t), u(t)), \tag{1.2}$$

such that the differential state derivatives $\dot{x}(t)$ are defined explicitly. A set of differential equations (1.1) in combination with an initial condition $x(t_0) = \hat{x}_0$ is often referred to as an initial value problem (IVP)

$$0 = f(\dot{x}(t), x(t), u(t)), \quad x(t_0) = \hat{x}_0, \tag{1.3}$$

given a control trajectory $u(t)$ over a certain time interval $t \in [t_0, t_\mathrm{f}]$. The following well-known theorem [69] then describes the existence and uniqueness of a solution $x(t)$ of the IVP in Eq. (1.3).

Theorem 1.2 (Picard-Lindelöf Theorem) *Consider the initial value problem from Eq.* (1.3) *for which the Jacobian matrix* $\frac{\partial f}{\partial \dot{x}}(\cdot)$ *is assumed to be invertible, corresponding to the ODE system as described in Definition 1.1, and where the continuous-time system dynamics* $f(\cdot)$ *are Lipschitz continuous in* $x(t)$ *and continuous in the control inputs* $u(t)$*. Then, for some value* $\epsilon > 0$*, there exists a unique solution* $x(t)$ *to the IVP on the interval* $[t_0 - \epsilon, t_0 + \epsilon]$*.*

This result will be important for direct optimal control methods, which typically rely on a piecewise polynomial control parameterization. The theorem can then be applied on each interval where the control inputs $u(t)$ remain continuous, corresponding to a piecewise formulated initial value problem. Note that stronger assumptions will typically be made throughout this thesis on the smoothness of the system dynamics $f(\cdot)$, especially in the context of sensitivity analysis starting with Chapter 2.

1.1.2 Differential-Algebraic Equations

Many modeling techniques do not directly result in an ODE formulation for the dynamic system as in Definition 1.1, but instead result in a set of DAE equations. An interesting example of such an approach for multi-body systems is based on Lagrange mechanics [148, 242, 248], which can be used to directly obtain the DAE system from the Lagrange functions and the algebraic constraints.

Definition 1.3 (Differential-Algebraic Equations) *We introduce the differential states* $x(t) \in \mathbb{R}^{n_\mathrm{x}}$*, control inputs* $u(t) \in \mathbb{R}^{n_\mathrm{u}}$ *and the algebraic variables* $z(t) \in \mathbb{R}^{n_\mathrm{z}}$ *to define the following set of Differential-Algebraic Equations (DAE) to describe the behaviour of a dynamic system*

$$0 = f(\dot{x}(t), x(t), z(t), u(t)). \tag{1.4}$$

If the Jacobian matrix $\frac{\partial f}{\partial z, \dot{x}}(\cdot)$ *is invertible, the DAE is said to be of index* 1*.*

The definition corresponds to a fully implicit formulation of a DAE system [65, 66]. It is additionally important to introduce the following common semi-explicit formulation of a DAE system:

$$\begin{aligned} \dot{x}(t) &= f_\mathrm{e}(x(t), z(t), u(t)) \\ 0 &= g(x(t), z(t), u(t)), \end{aligned} \tag{1.5}$$

where the function $g(\cdot)$ defines the algebraic equations. Note that often a relaxed formulation of these algebraic equations is used for dynamic optimization, on which more information can be found in [46, 175, 208]. The DAE system in (1.5) is of index 1 when the Jacobian matrix $\frac{\partial g}{\partial z}(\cdot)$ is invertible. We refer to the *differential index* or *differentiation index* whenever we mention the index of a DAE system throughout this thesis.

Definition 1.4 (Differential index) *The index [281] of the DAE in* (1.4) *is the minimum value for i such that the following set of differential equations*

$$0 = \frac{\mathrm{d}^i}{\mathrm{d}t^i} f(\dot{x}(t), x(t), z(t), u(t)),$$

corresponds to an ODE system as described in Definition 1.1.

Even though many interesting processes are naturally described by higher index DAEs, we assume all systems in this thesis to be of differential index 1. The class of models based on the Euler-Lagrange equations [242], for example, typically corresponds to DAE systems of index 3 in case of constrained dynamics described by non-minimal coordinates [248]. In practice, one often reformulates higher index DAEs into systems of index 1 or even ODE systems by use of index reduction techniques [241]. The Pantelides algorithm is based on a time differentiation of the constraint equations to obtain an index-1 DAE with associated consistency conditions. The resulting index-reduced system typically retains a reasonable symbolic complexity [148].

Remark 1.5 *A third interesting type of systems consists of Partial Differential Equations (PDE), which can depend on multiple other space variables in addition to the time dependency [26]. Note that ODE systems form a special case of DAEs, which in their turn form a special case of PDE systems. In addition, numerical methods for PDEs often rely on a particular discretization technique such as the Method of Lines (MOL) [302]. This particular approach performs the discretization first in space and then in time, resulting in a large structured set of ODEs or DAEs. More information on PDE-constrained optimization and direct optimal control for PDE systems can be found in [37, 167, 252] and references therein.*

Remark 1.6 *A final type consists of Delay Differential Equations (DDE) to describe time-delay systems [98]. They are not further considered in this thesis. Many practical applications involve actuators, sensors or communication networks that result in such delays. It is often crucial to take these delays into account within the control design [224].*

1.1.3 Additional Model Parameters

It is often the case that system dynamics are influenced by time-constant model parameters $p \in \mathbb{R}^{n_\mathrm{p}}$ such that the model function would instead be defined as $f(\dot{x}(t), x(t), u(t), p)$ in the case of an implicit ODE formulation. It is possible to reformulate such a system into the form of Definition 1.1 by extending the state dimension with additional variables that satisfy

$$\dot{x}_{n_\mathrm{x}+i}(t) = 0, \quad x_{n_\mathrm{x}+i}(t_0) = p_i, \quad i = 1, \ldots, n_\mathrm{p}.$$

In practice, the system dynamics could also be time-dependent $f(t, \dot{x}(t), x(t), u(t))$ which can be considered a special case of including a time-variable model parameter. In a similar fashion, this can be transformed in our standard formulation by defining the following additional state variable

$$\dot{x}_{n_\mathrm{x}+1}(t) = 1, \quad x_{n_\mathrm{x}+1}(t_0) = t_0.$$

Finally, it is interesting to note that an IVP has been introduced over a certain time interval $t \in [t_0, t_\mathrm{f}]$ in Eq. (1.3). Based on the definition $t(\tau) := t_0 + \tau T$ where $T = t_\mathrm{f} - t_0$ denotes the time horizon, the system dynamics could however be rescaled in time by using

$$\frac{\mathrm{d}x}{\mathrm{d}\tau} = \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau}, \quad \text{or} \quad x'(\tau) = T\,\dot{x}(t), \quad \text{or} \quad \dot{x}(t) = \frac{1}{T}\,x'(\tau),$$

resulting in the following scaled dynamics

$$0 = f\left(\frac{1}{T}\,x'(\tau), x(\tau), u(\tau)\right), \quad \tau \in [0, 1].$$

In this case, the time horizon could be either a constant $T = t_\mathrm{f} - t_0$ or a model parameter as introduced earlier. Note that the above reformulations are introduced mainly for notational convenience, while a separate treatment of these model parameters within the presented algorithmic techniques can typically be computationally more efficient.

1.1.4 Numerical Simulation

Based on the results of Theorem 1.2, we know that a unique state trajectory $x(t)$ exists as a solution to the IVP in Eq. (1.3). Note that an extension of this existence and uniqueness theorem can be made to DAE systems such as those in Definition 1.3. More information on this topic can, for example, be found in [279, 280]. In what follows, we often refer to the simulation of differential equations as a numerical approximation of this solution at discrete time points.

An overview on numerical simulation methods can be found in Chapter 2 based on [56, 157, 158]. But we already introduce a few basic concepts in this domain. Let $s_i(t_{i-1}, x_{i-1})$ denote the numerical approximation of the solution $x(t_i)$ at time point t_i, given the state values x_{i-1} at time point t_{i-1}. In that case, $s_{i-1}(t_{i-1}, x_{i-1}) = x_{i-1}$ holds by definition.

Definition 1.7 (Numerical integration error) *The local error for the numerical simulation result is defined as*

$$e(t_i) = x(t_i) - s_i(t_{i-1}, x(t_{i-1})),$$

where $T_{\text{int}} = t_i - t_{i-1}$ is typically referred to as the integration step size[1]*. The global or transported error is accordingly defined as*

$$E(t_i) = x(t_i) - s_i(t_0, x(t_0)),$$

which describes how errors are propagated by the integration scheme.

Based on these error definitions, let us additionally introduce the properties of convergence and order for a numerical integration scheme.

Definition 1.8 (Order of accuracy) *A numerical method is said to be convergent when its values converge to the exact solution for $T_{\text{int}} \to 0$, where T_{int} represents the integration step size. The method has order P if the local error satisfies*

$$\lim_{T_{\text{int}} \to 0} e(t_i) = \mathcal{O}(T_{\text{int}}^{P+1}).$$

Note that $P > 0$ is a necessary condition for convergence.

1.2 Direct Optimal Control

NMPC is an approach of increasing popularity for real-time control due to the ability to explicitly handle constraints and nonlinear dynamics that characterize the system of interest. This approach is based on the solution of a nonlinear Optimal Control Problem (OCP) at each sampling instant. Numerical methods to solve OCPs are typically classified into three main categories:

- *Dynamic Programming* (DP) schemes are based on Bellman's principle of optimality in order to propagate a cost-to-go function backwards in time [29]. This approach can either result in the continuous time Hamilton-Jacobi-Bellman (HJB) [214] partial differential equations or in a discrete time dynamic programming recursion [32].

[1] Unlike the standard notation in [157], the symbol T_{int} is used to denote the integration step size to avoid confusion, e.g., in the optimal control problem parameterization.

- *Indirect* methods for optimal control, also known as *first-optimize-then-discretize* approaches are typically based on Pontryagin's maximum principle [251] to formulate the infinite dimensional first-order necessary optimality conditions. The resulting nonlinear multi-point boundary value problem (BVP) can then be solved numerically [42, 239].
- *Direct* methods for optimal control, also known as *first-discretize-then-optimize* approaches are based on the numerical solution of a finite dimensional optimization problem which corresponds to a discrete approximation of the original continuous-time OCP. It can be shown that this direct approach converges to the original continuous-time solution as the discretization error decreases [119].

There are multiple difficulties when applying an indirect approach in practice, such as the conditioning of the resulting BVP [34]. In addition, direct methods can be implemented in a rather flexible manner while expert knowledge is typically needed whenever the problem formulation is changed within the indirect approach. This work therefore focuses on direct methods for real-time optimal control. The main advantage of dynamic programming is that the globally optimal solution can be found, unlike for the other two schemes. In general, this however comes at the cost of performing a tabulation of the cost-to-go function on a state space discretization grid. The DP approach therefore suffers from the *curse of dimensionality* [29] such that it can be applied to systems with only a few states, except for special cases. A more elaborate comparison of all three approaches can be found in [34, 59].

1.2.1 Parametric Optimal Control Problem

This thesis aims at solving parametric optimal control problems (OCPs) as they typically arise in Model Predictive Control (MPC) and Moving Horizon Estimation (MHE), on which more information can be found in Section 1.5. We introduce the following standard problem formulation

$$\min_{x(\cdot),\,u(\cdot)} \quad \int_0^T \ell(x(t),u(t))\,\mathrm{d}t + m(x(T)) \tag{1.6a}$$

$$\text{s.t.} \quad 0 = x(0) - \hat{x}_0, \tag{1.6b}$$

$$0 = f(\dot{x}(t), x(t), z(t), u(t)), \qquad \forall t \in [0,T], \tag{1.6c}$$

$$0 \geq h(x(t), u(t)), \qquad \forall t \in [0,T], \tag{1.6d}$$

$$0 \geq r(x(T)), \tag{1.6e}$$

where $x(t) \in \mathbb{R}^{n_x}$ denote the differential states, $\dot{x}(t)$ are the differential state derivatives, $z(t) \in \mathbb{R}^{n_z}$ are the algebraic variables and $u(t) \in \mathbb{R}^{n_u}$ denote the control inputs at time t. The objective in Eq. (1.6a) consists of the functions $\ell(\cdot)$ and $m(\cdot)$, which typically are referred to as respectively a Lagrange and a Mayer-type objective term. The optimization problem depends on the parameter value $\hat{x}_0$ through the initial value condition of Eq. (1.6b). The nonlinear dynamics in Eq. (1.6c) are described by an implicit DAE system of index 1, based on Definition 1.3. Additionally, Eqs. (1.6d) and (1.6e) denote respectively the path and terminal inequality constraints.

For the sake of simplicity, the algebraic variables only enter the DAE equations (1.6c) in the proposed continuous-time OCP formulation. In practice, both the objective and constraint functions can however additionally depend on these variables and a more general presentation of direct methods for DAE-constrained optimal control can be found in [50, 131, 208]. The time horizon T will typically be considered constant throughout this thesis, even though many interesting OCP problems rely on a variable end time such as, for example, in time-optimal formulations. Note that this can however be transformed in our standard formulation (1.6) based on a rescaling of the dynamics, presented in the previous section. We are mainly interested in the solution $u^*(t, \hat{x}_0)\ \forall t \in [0, T]$, which denotes a locally optimal control trajectory to be applied as a function of the current system state $\hat{x}_0$.

1.2.2 Multiple Shooting Parameterization

The continuous-time OCP formulation from (1.6) leaves us with an infinite dimensional optimization problem, which can be tackled using either one of the approaches mentioned earlier. We consider direct optimal control methods based on a parameterization of the state and control trajectory, resulting in a Nonlinear Programming (NLP) formulation which denotes a discrete-time approximation of the original problem. In this chapter, we focus on the direct multiple shooting method which was originally proposed as an approach to solve two-point BVPs [226, 239]. The idea was later adopted by Bock and Plitt into a direct method to solve optimal control problems [51]. Let us introduce the different ingredients for this problem parameterization.

Multiple shooting grid We first define a grid of $N + 1$ fixed control discretization points

$$0 = t_0 < t_1 < \ldots < t_N = T,$$

which partition the control horizon into N shooting intervals. For the sake of simplicity, we consider here an equidistant grid over the control horizon

consisting of the collection of time points t_i, where $t_{i+1} - t_i = \frac{T}{N} =: T_\mathrm{s}$ for $i = 0, \ldots, N-1$. Note however that a non-equidistant grid can be treated in a similar fashion within direct multiple shooting.

Control parameterization Next, one needs to decide on a parameterization of the control trajectory $u(t)\ \forall t \in [0, T]$ by a finite number of parameters which yield a suitable approximation. Typically, one relies on a piecewise polynomial approximation given the N shooting intervals of the control discretization grid [209]. For notational convenience, we use the simplest form based on a piecewise constant parameterization

$$u(\tau) = u_i \quad \text{for} \quad \tau \in [t_i, t_{i+1}).$$

State parameterization The multiple shooting grid and the control parameterization result in a natural representation of the state trajectory as the solution to N coupled initial value problems

$$0 = f(\dot{x}(\tau), x(\tau), z(\tau), u_i), \quad x(t_i) = x_i, \quad \tau \in [t_i, t_{i+1}), \tag{1.7}$$

for $i = 0, \ldots, N-1$ and $x(0) = \hat{x}_0$ is defined. When the initial condition for $x(t_i)$ refers to the solution of the previous IVP, one obtains a sequential approach to direct optimal control which is typically referred to as *single shooting* [288]. This method relies on a sequential evaluation of the full state trajectory, where the solution of each IVP depends on the simulation result of the previous one. Instead, the direct multiple shooting method introduces additional state variables $x_i \in \mathbb{R}^{n_\mathrm{x}}$ for $i = 0, \ldots, N$ on the grid points in order to explicitly couple the initial value problems. Note that this method is consistent with the sequential approach by including the continuity constraints

$$x_{i+1} = s_{i+1}(t_i, x_i, u_i), \quad i = 0, \ldots, N-1,$$

where $s(\cdot)$ denotes a numerical simulation for the IVP (1.7) given the value $x(t_i) = x_i$. For notational convenience, we further consider the same integration method to be used to solve the IVP on each interval of the equidistant multiple shooting grid. We therefore rewrite these conditions as

$$x_{i+1} = \phi(x_i, u_i), \quad i = 0, \ldots, N-1,$$

resulting in a discrete-time and autonomous representation of the system dynamics. Throughout this thesis, we refer to the function $\phi(\cdot)$ as the result of calling an *integrator*.

Path inequality constraints The inequality constraints (1.6d) need to be imposed on the full time horizon $[0, T]$, resulting in a semi-infinite programming problem [165] which is generally not tractable. More information on how to practically avoid violations of these semi-infinite path constraints within direct optimal control methods can, for example, be found in [254]. For notational convenience, we simply relax these path constraints by imposing them directly on the shooting grid points

$$0 \geq h(x_i, u_i), \quad i = 0, \ldots, N-1,$$

which in practice often already limits the violations of these constraints in between the grid points of a sufficiently fine discretization.

Nonlinear Programming (NLP) formulation To arrive at the finite dimensional NLP approximation of the OCP, we additionally approximate the continuous time objective (1.6a) by the discrete time sum

$$\sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N),$$

which can be obtained, for example, by extending the dynamics (1.6c) with quadrature states [166]. The discrete time stage cost can be evaluated efficiently as part of the solution of the state trajectory IVP [208].

Based on the above ingredients for parameterization of the OCP in Eq. (1.6), the resulting structured Nonlinear Program (NLP) reads

$$\min_{X,\,U} \quad \sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) \tag{1.8a}$$

$$\text{s.t.} \quad 0 = x_0 - \hat{x}_0, \tag{1.8b}$$

$$0 = x_{i+1} - \phi(x_i, u_i), \quad i = 0, \ldots, N-1, \tag{1.8c}$$

$$0 \geq h(x_i, u_i), \quad i = 0, \ldots, N-1, \tag{1.8d}$$

$$0 \geq r(x_N), \tag{1.8e}$$

with state trajectory $X = [x_0^\top, \ldots, x_N^\top]^\top$ where $x_i \in \mathbb{R}^{n_x}$ and control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$ where $u_i \in \mathbb{R}^{n_u}$. When one addresses this problem directly in a nonlinear optimization framework, the variables in X and U generally represent a feasible state and control trajectory only at convergence. This is additionally illustrated in Figure 1.1, which visualizes the converged and unconverged multiple shooting trajectories.

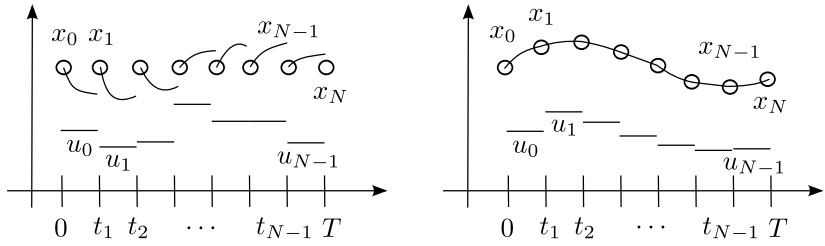


Figure 1.1: Illustration of the unconverged (left) and converged (right) state and control trajectories of a multiple shooting method.

Remark 1.9 *Direct multiple shooting is often referred to as a simultaneous approach because the optimization and simulation problem are solved together. In comparison, a sequential approach carries out the simulation task separately from solving the optimization problem. This technique is also known as single shooting [288], where a reduced OCP formulation is obtained after replacing the variables x_i by the results of a forward simulation* (1.7)*. Since the variable space of this problem is strongly reduced in dimension from $(N+1)n_{\mathrm{x}} + Nn_{\mathrm{u}}$ to only Nn_{u}, the task of solving this NLP appears to be simplified. However, it has been shown [8] that the cost per Newton iteration can be made equal for both approaches because of the sparsity structure in* (1.8)*. Advantages of multiple shooting over single shooting are the stronger flexibility in initializing the problem and parallelizing the algorithm, and the improved convergence properties especially in the case of an unstable system [8].*

1.2.3 Direct Transcription Methods

Direct transcription methods formulate an optimization problem where the equations to simulate the system dynamics are embedded as nonlinear equality constraints [34, 250]. This is therefore an alternative simultaneous approach for direct optimal control. These methods have the same property of representing a feasible state and control trajectory only at convergence, as illustrated by Figure 1.1 for multiple shooting. More specifically, this approach requires that all intermediate results for numerical simulation need to be *lifted* as additional degrees of freedom [8]. A rather popular example of such a direct transcription method is direct collocation [39, 160], where the additional constraints and variables are based on the equations to define the corresponding collocation polynomials. This family of collocation methods is discussed for numerical simulation as part of Chapter 2. The efficient solution of the resulting direct collocation NLP will be the topic of discussion in Chapter 6.

Let us introduce the following general representation of a numerical simulation method applied to one shooting interval

$$x_{i+1} = F(x_i, z_i, u_i)$$
$$0 = G(x_i, z_i, u_i),$$

where z_i for $i = 0, \ldots, N-1$ denote all intermediate variables of the integration scheme to solve the IVP (1.7). They are defined either explicitly or implicitly by the function $G(\cdot)$ and they are necessary to evaluate the numerical simulation result $x_{i+1} = F(x_i, z_i, u_i)$. The resulting structured NLP reads as

$$\min_{X,\,U,\,Z} \quad \sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) \tag{1.9a}$$

$$\text{s.t.} \quad 0 = x_0 - \hat{x}_0, \tag{1.9b}$$

$$0 = x_{i+1} - F(x_i, z_i, u_i), \quad i = 0, \ldots, N-1, \tag{1.9c}$$

$$0 = G(x_i, z_i, u_i), \quad i = 0, \ldots, N-1, \tag{1.9d}$$

$$0 \geq h(x_i, u_i), \quad i = 0, \ldots, N-1, \tag{1.9e}$$

$$0 \geq r(x_N). \tag{1.9f}$$

Even though this NLP formulation is much larger than the multiple shooting type problem (1.8), direct transcription methods have the important property that the sparsity of the dynamics can typically be preserved within the additional constraints (1.9c) and (1.9d). In addition, the solution of the augmented NLP formulation in Eq. (1.9) relies on the evaluation of the functions $F(\cdot)$ and $G(\cdot)$ and their respective derivatives. For direct multiple shooting, one needs to propagate the sensitivities directly through the dynamics as discussed in the next chapter. On the other hand, multiple shooting can rely on any (general-purpose) solver to simulate the system dynamics. Many of these solvers include step size and order control to provide guarantees regarding the accuracy of the simulation results, which typically allows a reduced overall number of integration steps [158]. See [7, 27, 166] for more details about the use of step size control, especially within direct optimal control. Alternatively, it is less trivial but not impossible to extend the direct transcription formulation in Eq. (1.9) with such an adaptive approach [36, 215, 246]. A more detailed comparison between the multiple shooting method and the direct collocation approach can be found as part of Chapter 6 on lifted collocation integrators.

1.3 Nonlinear Programming Methods

This section introduces a few important concepts and algorithmic techniques from the field of nonlinear optimization, with a focus on the tools needed for real-time optimal control applications. A detailed overview can be found in [39, 112, 232] and references therein. For this purpose, we consider a compact Nonlinear Programming (NLP) formulation that covers both optimal control problems in Eqs. (1.8) and (1.9). All optimization variables are collected into the vector $y \in \mathbb{R}^{n_y}$ such that the standard NLP form reads

$$\min_{y} \quad \psi(y) \tag{1.10a}$$

$$\text{s.t.} \quad 0 \;=\; c(y) \tag{1.10b}$$

$$0 \;\geq\; h(y), \tag{1.10c}$$

where $\psi : \mathbb{R}^{n_y} \to \mathbb{R}$, $c : \mathbb{R}^{n_y} \to \mathbb{R}^{n_{eq}}$, $h : \mathbb{R}^{n_y} \to \mathbb{R}^{n_{ineq}}$. Because of the numerical methods presented in this section and also further in this thesis, we introduce the following assumption on all problem functions.

Assumption 1.10 *The objective $\psi(y)$ and constraint functions $c(y)$ and $h(y)$ in the NLP from Eq.* (1.10) *are twice continuously differentiable in all arguments.*

1.3.1 Optimality Conditions

We are interested in the conditions for optimality of the NLP in Eq. (1.10), which is not necessarily convex in general. Because of our focus on real-time feasible implementations of direct optimal control methods, it will typically be sufficient to find a locally optimal solution.

Definition 1.11 (Feasible point) *A vector $\bar{y}$ is called a feasible point, iff the constraints in Eqs.* (1.10b) *and* (7.21c) *are satisfied*

$$0 = c(\bar{y}), \quad 0 \geq h(\bar{y}).$$

The feasible set Ω is then defined to include all such points, i.e., $\Omega = \{y \in \mathbb{R}^{n_y} \,|\, 0 = c(y), 0 \geq h(y)\}$.

Definition 1.12 (Local minimizer) *A vector $y^\star$ is called a local minimizer or a locally optimal solution, iff this point is feasible $y^\star \in \Omega$ and there exists a local neighborhood $\mathcal{N}$ of $y^\star$, i.e., an open ball around $y^\star$, such that*

$$\forall y \in \Omega \,\cap\, \mathcal{N} : \psi(y) \geq \psi(y^\star). \tag{1.11}$$

A local minimizer can also be the global solution when the latter condition holds for the complete feasible set Ω instead of a local neighborhood $\mathcal{N}$. Even though many recent developments have been made with respect to algorithms for global optimization [113, 174], they are generally not yet tractable especially for real-time optimal control of nontrivial systems with relatively fast dynamics. The online optimization algorithms in this thesis are even based on the approximative solution of the optimization problem at one sampling time, starting from the approximative solution of the previous problem. Based on a good initialization and a sufficiently high sampling rate, these methods result in a local *convergence over time* as will be discussed in Section 1.5.

In order to present the optimality conditions that characterize a local minimizer, we first need to introduce a few more concepts.

Definition 1.13 (Active constraint) *An inequality constraint $h_i : \mathbb{R}^{n_y} \to \mathbb{R}$ is called active in a feasible point $\bar{y} \in \Omega$, iff*

$$h_i(\bar{y}) = 0.$$

The index set $\mathcal{A}(\bar{y}) \subset \{1, \ldots, n_{\text{ineq}}\}$ of active inequality constraints is typically referred to as the active set.

Definition 1.14 (LICQ) *The linear independence constraint qualification (LICQ) holds at a feasible point $\bar{y} \in \Omega$, iff all vectors $\nabla c_i(\bar{y})$ for $i \in \{1, \ldots, n_{\text{eq}}\}$ and $\nabla h_i(\bar{y})$ for $i \in \mathcal{A}(\bar{y})$ are linearly independent.*

Note that weaker constraint qualifications exist as discussed in [232], even though the above LICQ condition can often be satisfied in practice and will therefore be sufficient for our purposes. Let us introduce the Lagrangian function, which plays an important role in the optimality conditions.

Definition 1.15 (Lagrangian function) *For the NLP formulation in Eq.* (1.10)*, the Lagrangian function is defined as*

$$\mathcal{L}(y, \lambda, \nu) = \psi(y) + \lambda^\top c(y) + \nu^\top h(y). \tag{1.12}$$

The vectors $\lambda \in \mathbb{R}^{n_{\text{eq}}}$ and $\nu \in \mathbb{R}^{n_{\text{ineq}}}$ denote the Lagrange multipliers or the dual variables, which can be interpreted as shadow prices [232].

The following theorem then specifies the first order necessary conditions (FONC) to characterize a local minimizer.

Theorem 1.16 (Karush-Kuhn-Tucker (KKT) conditions) *Let $y^\star$ be a local minimizer for the NLP in* (1.10) *which satisfies the LICQ condition. Then,*

there exist Lagrange multipliers $\lambda^\star$ and $\nu^\star$ for which holds

$$\nabla_y \mathcal{L}: \quad \nabla\psi(y^\star) + \nabla c(y^\star)\lambda^\star + \nabla h(y^\star)\nu^\star = 0 \tag{1.13a}$$

$$\nabla_\lambda \mathcal{L}: \quad c(y^\star) = 0 \tag{1.13b}$$

$$\nabla_\nu \mathcal{L}: \quad h(y^\star) \leq 0 \tag{1.13c}$$

$$\nu^\star \geq 0 \tag{1.13d}$$

$$\nu_i^\star h_i(y^\star) = 0, \quad i = 1, \ldots, n_{\text{ineq}}. \tag{1.13e}$$

A primal-dual solution $(y^\star, \lambda^\star, \nu^\star)$ is called a KKT point.

Note that we use the notation $\frac{\partial c}{\partial y}(\bar{y}) = \nabla c(\bar{y})^\top = c_y(\bar{y})$ to denote a Jacobian. When the LICQ holds at the local minimizer $y^\star$, the optimal Lagrange multipliers $(\lambda^\star, \nu^\star)$ are unique. A proof for this and Theorem 1.16 can be found in [232]. Eqs. (1.13b) and (1.13c) are typically referred to as the *primal feasibility* conditions, while Eq. (1.13a) denotes the *stationarity* conditions, Eq. (1.13d) states the *dual feasibility* and Eq. (1.13e) forms the *complementarity* conditions. In the case of a convex NLP, every local minimizer is also a global solution and these first order KKT conditions are then both necessary and sufficient. For the general non-convex form, we introduce the second order sufficient conditions (SOSC) after defining strict complementarity.

Definition 1.17 (Strict complementarity) *Let $(y^\star, \lambda^\star, \nu^\star)$ be a KKT point of the NLP in* (1.10)*. Strict complementarity then holds for this solution if all active inequality constraints are strictly active, i.e., if $\nu_i^\star > 0$ for all $i \in \mathcal{A}(y^\star)$.*

In order to state the second-order sufficient conditions, we additionally define the critical cone at a KKT point.

Definition 1.18 (Critical cone) *Let $(y^\star, \lambda^\star, \nu^\star)$ be a KKT point of the NLP in* (1.10)*, then the critical cone corresponds to the set*

$$d \in \mathcal{C}(y^\star, \nu^\star) \Leftrightarrow \begin{cases} \nabla c_i(y^\star)^\top d = 0, & \forall i \in \{1, \ldots, n_{\text{eq}}\} \\ \nabla h_i(y^\star)^\top d = 0, & \forall i \in \mathcal{A}(y^\star) \text{ with } \nu_i^\star > 0 \\ \nabla h_i(y^\star)^\top d \leq 0, & \forall i \in \mathcal{A}(y^\star) \text{ with } \nu_i^\star = 0. \end{cases} \tag{1.14}$$

Theorem 1.19 (Second-order sufficient conditions) *Let $(y^\star, \lambda^\star, \nu^\star)$ be a KKT point of the NLP in* (1.10)*. If the Hessian of the Lagrangian is strictly positive definite in the directions of the critical cone, i.e., if the following condition holds*

$$d^\top \nabla_y^2 \mathcal{L}(y^\star, \lambda^\star, \nu^\star)\, d > 0, \quad \forall d \in \mathcal{C}(y^\star, \nu^\star),\ d \neq 0,$$

then the point $y^\star$ is a local minimizer of (1.10)*.*

We then define a regular KKT point, based on the collection of conditions which we assume throughout this thesis, to simplify our discussion.

Definition 1.20 (Regular KKT point) *A local minimizer of the constrained NLP in* (1.10) *is called a regular KKT point* $(y^\star, \lambda^\star, \nu^\star)$ *if LICQ, strict complementarity and SOSC are satisfied at this point.*

Assumption 1.21 *The local minimizers of the optimal control problems in Eqs.* (1.8) *and* (1.9) *are assumed to be regular KKT points.*

1.3.2 Newton-Type Optimization

An overview of algorithms for nonlinear optimization can be found in [39, 112, 232]. In this thesis, we only consider Newton-type optimization algorithms to solve the NLP (1.10) with a focus on Sequential Quadratic Programming (SQP). In order to discuss optimization algorithms and their convergence properties, we briefly introduce some of the common convergence rates [232].

Definition 1.22 (q-linear convergence rate) *Let* $y^{(k)} \in \mathbb{R}^{n_y}$ *denote the iterates where* $y^{(k)} \to y^\star$ *for* $k \to \infty$*. We say that* $y^{(k)}$ *converges q-linearly if there exists a constant* $r \in (0,1)$ *such that*

$$\frac{\|y^{(k+1)} - y^\star\|}{\|y^{(k)} - y^\star\|} \leq r, \qquad \forall k \geq \bar{k} \in \mathbb{N}.$$

Definition 1.23 (q-superlinear convergence rate) *Let* $y^{(k)} \in \mathbb{R}^{n_y}$ *denote the iterates where* $y^{(k)} \to y^\star$ *for* $k \to \infty$*. We say* $y^{(k)}$ *converges q-superlinearly if*

$$\lim_{k\to\infty} \frac{\|y^{(k+1)} - y^\star\|}{\|y^{(k)} - y^\star\|} = 0.$$

Definition 1.24 (q-quadratic convergence rate) *Let* $y^{(k)} \in \mathbb{R}^{n_y}$ *denote the iterates where* $y^{(k)} \to y^\star$ *for* $k \to \infty$*. We say that* $y^{(k)}$ *converges q-quadratically if there exists a constant* $K \in (0, \infty)$ *such that*

$$\frac{\|y^{(k+1)} - y^\star\|}{\|y^{(k)} - y^\star\|^2} \leq K, \qquad \forall k \geq \bar{k} \in \mathbb{N}.$$

Equality constrained optimization

Let us first consider the equality constrained case, i.e., $n_{\text{ineq}} = 0$ and $\mathcal{L}(y, \lambda) = \psi(y) + \lambda^\top c(y)$. Newton-type optimization algorithms are based on the direct

solution of the first order necessary conditions

$$\begin{aligned} \nabla_y \mathcal{L}(y,\lambda) &= 0 \\ c(y) &= 0, \end{aligned}$$

from Theorem 1.16 and this using a Newton-type root finding method [79, 82]. An exact Newton iteration on these KKT conditions reads as

$$\underbrace{\begin{bmatrix} \nabla_y^2 \mathcal{L}(\bar{y},\bar{\lambda}) & c_y^\top(\bar{y}) \\ c_y(\bar{y}) & \mathbb{0} \end{bmatrix}}_{=J(\bar{y},\bar{\lambda})} \begin{bmatrix} \Delta y \\ \Delta\lambda \end{bmatrix} = - \underbrace{\begin{bmatrix} \nabla_y \mathcal{L}(\bar{y},\bar{\lambda}) \\ c(\bar{y}) \end{bmatrix}}_{=\mathcal{F}(\bar{y},\bar{\lambda})}, \tag{1.15}$$

where $c_y(\bar{y}) := \frac{\partial c}{\partial y}(\bar{y}) = \nabla c(\bar{y})^\top$ denotes the Jacobian matrix and the values $\bar{y}$ and $\bar{\lambda}$ denote the current primal and dual variables. To arrive at a more compact notation, we refer to the exact Newton iteration as $J(\bar{y},\bar{\lambda}) \begin{bmatrix} \Delta y \\ \Delta\lambda \end{bmatrix} = -\mathcal{F}(\bar{y},\bar{\lambda})$ based on the Jacobian matrix $J(\bar{y},\bar{\lambda}) := \frac{\partial \mathcal{F}}{\partial(y,\lambda)}(\bar{y},\bar{\lambda})$.

There exist many Newton-type optimization methods that are based on specific approximations of the KKT matrix $J(\cdot)$ in Eq. (1.15), in order to result in desirable convergence properties at a considerably reduced computational cost. Such an approach can be based on the approximation of the Hessian of the Lagrangian $\tilde{H} \approx H := \nabla_y^2 \mathcal{L}(\bar{y},\bar{\lambda})$ using only first order derivative information as performed in the family of Quasi-Newton methods [80, 232]. A compact overview of Hessian approximation schemes will be provided further in the context of SQP methods. Other Newton-type optimization algorithms even use an inexact Jacobian matrix for the equality constraints [49, 94, 175], as will be the topic of discussion in Chapter 7.

One iteration of any such inexact Newton-type method can be written as follows in the compact form

$$\tilde{J}(\bar{y},\bar{\lambda}) \begin{bmatrix} \Delta y \\ \Delta\lambda \end{bmatrix} = -\mathcal{F}(\bar{y},\bar{\lambda}), \tag{1.16}$$

based on a Jacobian approximation $\tilde{J}(\bar{y},\bar{\lambda}) \approx J(\bar{y},\bar{\lambda}) := \frac{\partial \mathcal{F}}{\partial(y,\lambda)}(\bar{y},\bar{\lambda})$ and where the function $\mathcal{F}(\cdot)$ denotes the KKT right-hand side in Eq. (1.15). The convergence of this scheme then follows the classical and well-known local contraction result from [44, 82, 94, 252]. We use a particular version of this theorem from [86], providing sufficient and necessary conditions for the existence of a neighborhood of the solution where the Newton-type iteration converges. For this purpose, we consider the full-step iteration

$$\bar{w}^+ = \bar{w} - \tilde{J}(\bar{w})^{-1} \mathcal{F}(\bar{w}), \tag{1.17}$$

where the current primal-dual iterate $\bar{w} := (\bar{y}, \bar{\lambda})$ is defined. We first present a classical result from nonlinear systems theory [218], which we do not prove here. Let $\rho(P)$ denote the spectral radius, i.e., the maximum absolute value of the eigenvalues for the square matrix P.

Lemma 1.25 (Linear stability analysis) *We regard an iteration of the form $\bar{w}^+ = \mathcal{G}(\bar{w})$ with $\mathcal{G}(\cdot)$ a continuously differentiable function in a neighborhood of a fixed point $w^\star = \mathcal{G}(w^\star)$. If the spectral radius of the Jacobian matrix is smaller than one, $\rho\left(\frac{\partial \mathcal{G}}{\partial w}(w^\star)\right) < 1$, then the fixed point is asymptotically stable. More specifically, the iterates converge to $w^\star$ with a q-linear convergence rate with asymptotic contraction factor $\rho\left(\frac{\partial \mathcal{G}}{\partial w}(w^\star)\right)$. On the other hand, if $\rho\left(\frac{\partial \mathcal{G}}{\partial w}(w^\star)\right) > 1$, then the fixed point is unstable and the iterations can move away, even if the initial guess is arbitrarily close to $w^\star$.*

We can apply this lemma to the Newton-type iteration $\mathcal{G}(w) = w - \tilde{J}(w)^{-1}\mathcal{F}(w)$, based on the extra assumption that the Jacobian approximation $\tilde{J}(\cdot)$ is continuously differentiable, in addition to being invertible. This assumption is satisfied for an exact Newton method, for schemes based on fixed Jacobian approximations as well as for the Gauss-Newton (GN) method for nonlinear least squares optimization, which will be introduced further. We can now present the main local contraction theorem.

Theorem 1.26 (Local Newton-type contraction) *We consider the twice continuously differentiable function $\mathcal{F}(y, \lambda)$ from Eq. (1.15) given Assumption 1.10 and the regular KKT point $\mathcal{F}(y^\star, \lambda^\star) = 0$. We apply the Newton-type iteration from Eq. (1.16), where the Jacobian approximation $\tilde{J}(\bar{y}, \bar{\lambda}) \approx J(\bar{y}, \bar{\lambda})$ is assumed to be continuously differentiable and invertible in a neighborhood of the solution. If all eigenvalues of the iteration matrix have a modulus smaller than one, i.e., if the spectral radius*

$$\kappa^\star := \rho\left(\tilde{J}(y^\star, \lambda^\star)^{-1} J(y^\star, \lambda^\star) - \mathbb{1}\right) < 1, \tag{1.18}$$

then this fixed point $(y^\star, \lambda^\star)$ is asymptotically stable. In addition, the iterates $(\bar{y}, \bar{\lambda})$ converge linearly to the KKT point $(y^\star, \lambda^\star)$ with the asymptotic contraction rate $\kappa^\star$ when initialized sufficiently close. On the other hand, if $\kappa^\star > 1$, then the fixed point $(y^\star, \lambda^\star)$ is unstable.

Proof. We apply Lemma 1.25 directly to the Newton-type iteration $\mathcal{G}(w) = w - \tilde{J}(w)^{-1}\mathcal{F}(w)$, where $w := (y, \lambda)$ denotes all the optimization variables. First, we observe that $w^\star = \mathcal{G}(w^\star)$ holds, due to $\mathcal{F}(w^\star) = 0$. We then compute

the Jacobian of $\mathcal{G}(\cdot)$ at the fixed point $w^\star$:

$$\begin{aligned}\frac{\partial \mathcal{G}}{\partial w}(w^\star) &= \mathbb{1} - \frac{\partial(\tilde{J}^{-1})}{\partial w}(w^\star)\underbrace{\mathcal{F}(w^\star)}_{=0} - \tilde{J}(w^\star)^{-1}\frac{\partial \mathcal{F}}{\partial w}(w^\star) \\ &= \mathbb{1} - \tilde{J}(w^\star)^{-1}J(w^\star),\end{aligned}$$

where $\mathbb{1}$ denotes the identity matrix. □

In summary, the spectral radius of the iteration matrix is a tight criterion for local Newton-type convergence. If it is larger than one, the Newton-type method diverges, while if it is smaller than one, the method converges. More advanced convergence results, under weaker assumptions, can be found in [77, 82, 206]. Note that the exact Newton method from Eq. (1.15) has a locally quadratic convergence rate [79, 82]. Even though Theorem 1.26 holds for any method that satisfies the corresponding conditions, other Newton-type variants exist that, for example, allow locally superlinear convergence [94, 143] based on quasi-Newton Jacobian updates. In the case of optimal control for differential-algebraic equations, even quadratic convergence rates have been shown for a specific Newton-type optimization method with inexact derivatives in [175].

Remark 1.27 *Results on global convergence will not be discussed throughout this thesis, even though a lot of research has focused on globalization strategies such as line search, trust-region or filter methods [39, 112, 232]. In the context of embedded optimization where a good initial guess is typically available from the previous iterate of the online algorithm, local convergence results and contraction rates instead become more valuable in practice [90].*

Inequality constrained optimization

In order to extend Newton-type optimization algorithms to the inequality constrained case, one needs to decide on a way to treat these inequality constraints in the KKT conditions (1.13). We briefly discuss two popular classes of algorithms for nonlinear optimization: Interior Point (IP) methods and Sequential Quadratic Programming (SQP).

Interior point or *barrier* methods are among the most powerful algorithms for nonlinear optimization, especially for large-scale programming problems [232]. The main idea is to approximate the non-smooth complementarity condition in Eq. (1.13e) by the smooth version $\nu_i h_i(y) = \tau$ for $i = 1, \ldots, n_{\text{ineq}}$ and where $\tau > 0$ is typically referred to as the barrier parameter. The resulting perturbed KKT system is subsequently solved for a sequence of positive barrier parameter

values for which $\tau \to 0$. The corresponding primal and dual solutions $\bar{y}(\tau)$, $\bar{\lambda}(\tau)$ and $\bar{\nu}(\tau)$ can then be shown to locally coincide with the *primal-dual central path*, which converges to the solution of the original KKT system (1.13). This approach can also be interpreted as solving the KKT conditions of a barrier problem that handles the inequality constraints by means of a barrier function, and this for a sequence of positive parameter values. As the name suggests, the iterates of an IP method remain in the strict interior of the feasible region even though an infeasible start is typically possible based on a slack reformulation. More information on this family of optimization algorithms can be found in [39, 232, 327]. A widely used general-purpose IP method is implemented for sparse NLPs in the open-source code `Ipopt` [317, 318].

Unlike the smoothening technique of interior point methods, Sequential Quadratic Programming (SQP) is based on sequentially approximating the NLP by a quadratic subproblem. The combinatorial complexity of determining the correct active set then needs to be addressed for each simpler subproblem. This approximation is typically done using a convex Quadratic Programming (QP) subproblem. But generalizations of this idea exist under the collective name of Sequential Convex Programming (SCP) as discussed in [304, 305], including Sequential Semidefinite Programming (SSDP) in [120, 300]. One important motivation for SQP methods is based on the observation that for a fixed active set, Newton iterations on the KKT optimality conditions (1.15) can be interpreted as solving equality constrained QP approximations of the original NLP [232]. Unlike IP techniques, which do not easily benefit from a good initial guess [296], the natural warm-starting capabilities of SQP methods make them especially popular for embedded optimization [90]. This thesis therefore focuses on the family of SQP methods for nonlinear programming and a compact overview is provided next.

1.3.3 Sequential Quadratic Programming

An SQP method applied to the general NLP in Eq. (1.10) proceeds in each iteration by solving the following QP subproblem

$$\min_{\Delta y} \quad \frac{1}{2}\Delta y^\top H \Delta y + \nabla_y \psi(\bar{y})^\top \Delta y \tag{1.19a}$$

$$\text{s.t.} \quad 0 \;=\; c(\bar{y}) + c_y(\bar{y})\Delta y \qquad \mid \lambda_{\text{QP}} \tag{1.19b}$$

$$0 \;\geq\; h(\bar{y}) + h_y(\bar{y})\Delta y \qquad \mid \nu_{\text{QP}}, \tag{1.19c}$$

where $c_y(\bar{y}) := \frac{\partial c}{\partial y}(\bar{y})$ and $h_y(\bar{y}) := \frac{\partial h}{\partial y}(\bar{y})$ denote the constraint Jacobian matrices. In the case of an exact SQP method, the matrix $H = \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\nu})$

denotes the Hessian of the Lagrangian. The current primal and dual variables are updated based on the solution of the QP (1.19) as follows

$$\bar{y}^+ = \bar{y} + \alpha \Delta y$$

$$\bar{\lambda}^+ = \bar{\lambda} + \alpha(\lambda_{\mathrm{QP}} - \bar{\lambda})$$

$$\bar{\nu}^+ = \bar{\nu} + \alpha(\nu_{\mathrm{QP}} - \bar{\nu}),$$

where the step size α is typically determined using a globalization strategy [232]. Throughout this thesis, we omit such globalization techniques in the context of online algorithms for embedded optimization (see Remark 1.27) and we therefore consider $\alpha = 1$. As mentioned earlier, we have a special focus on SQP methods for real-time optimal control applications because of their warm-starting capabilities. In addition, SQP methods satisfy linearized versions of the inequality constraints (1.19c) in each iteration which will show to be especially important within online algorithms for Nonlinear MPC in Section 1.5. In what follows, we provide a compact overview of Hessian approximation techniques and local convergence results for SQP methods.

Hessian approximation techniques

Exact Hessian based SQP methods rely on the computation of the second order derivatives $H = \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\nu})$ which can be computationally expensive in general [268]. Many popular variants of SQP methods are therefore based on Hessian approximation techniques that often result in a slower local convergence rate but this at a considerably lower computational cost per iteration, typically resulting in a reduction of the overall computation time. An important class of such SQP algorithms uses exact constraint Jacobians but approximates the Hessian matrix H, using an update formula based on first order derivative information [255]. These schemes are better known as Quasi-Newton methods and the most widely used formula is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [232]. Quasi-Newton methods are very successful for nonlinear optimization, e.g., in the form of the software packages `SNOPT` [133] and `MUSCOD-II` [209]. Based on our experience, these techniques are however less recommendable in the context of real-time optimal control due to their fluctuating convergence rate. For example, after a state jump due to external noise, the Hessian approximation can be poor such that many BFGS updates are needed to recover a reasonable convergence rate. This is in conflict with the real-time or deterministic run-time requirements as they will be specified in Section 1.5.

Another popular Hessian approximation technique is used in the Constrained or Generalized Gauss-Newton (GGN) method as presented in [48, 292]. This scheme is applicable to optimization problems with a (nonlinear) least squares type objective $\psi(y) = \frac{1}{2}\|R(y)\|_2^2$ which is quite common in optimal control problems. The function $R(\cdot)$ is often referred to as the residual function, since it typically denotes an error or misfit [48]. The GGN method approximates the Hessian of the Lagrangian using the matrix

$$H_{\mathrm{GN}}(\bar{y}) = R_y(\bar{y})^\top R_y(\bar{y}) \;\approx \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\nu}), \tag{1.20}$$

where $R_y(\bar{y}) := \frac{\partial R}{\partial y}(\bar{y})$ denotes the Jacobian of the residual at the current optimization values $\bar{y}$. Note that the Gauss-Newton Hessian approximation $H_{\mathrm{GN}}(y)$ does not depend on the dual variables and is therefore also called a *multiplier-free* approach. This is an important advantage in practice, because it means that its performance does not depend on the availability of a good initial guess for these Lagrange multipliers. The resulting objective (1.19a) in the QP subproblem reads as

$$\begin{aligned}\frac{1}{2}\|R(\bar{y}) + R_y(\bar{y})\Delta y\|_2^2 &= \frac{1}{2}\Delta y^\top R_y(\bar{y})^\top R_y(\bar{y})\Delta y + R(\bar{y})^\top R_y(\bar{y})\Delta y + \text{constant} \\ &= \frac{1}{2}\Delta y^\top H_{\mathrm{GN}}\Delta y + \nabla_y \psi(\bar{y})^\top \Delta y + \text{constant}.\end{aligned}$$

The GGN method is based on the observation that the matrix in Eq. (1.20) forms a good Hessian approximation, as long as the residual evaluations $R(\cdot)$ remain small or the problem functions have small second order derivatives [232]. This is typically the case in parameter estimation problems or tracking formulations of optimal control, which are two important application examples [48]. In case of a perfect fit, i.e., zero residuals at the solution, a locally quadratic convergence rate can even be observed. In all other situations, the GGN method results in a typically (fast) linear convergence rate.

Subproblem convexification

For a regular KKT point, based on the SOSC condition from Theorem 1.19, the Hessian of the Lagrangian is locally positive definite on the null space of the active constraints. This, however, does not mean that each QP subproblem (1.19) will be convex when initializing the SQP method further away from this KKT point. In general, *regularization* or *convexification* techniques are therefore used to result in convex QP subproblems [133, 134, 232]. Both of the aforementioned Hessian approximation techniques allow one to avoid this issue in a rather convenient way. The Gauss-Newton Hessian approximation is positive definite

by definition, if $\frac{\partial R}{\partial y}$ has full rank, and therefore does not require regularization. Quasi-Newton methods often rely on a modified update formula in order to maintain a positive definite Hessian approximation as discussed in [232]. In case of optimal control problems such as Eq. (1.8) for direct multiple shooting, it is important to apply an efficient convexification technique that additionally preserves the sparsity structure of the problem as proposed in [310].

Alternative inexact SQP methods

In addition to Quasi-Newton and Gauss-Newton methods, even fixed Hessian approximations are sometimes used in practice because of computational advantages. An SQP scheme with a fixed Hessian approximation is equivalent to a preconditioned proximal gradient method with fixed step size [230, 336] as discussed in [193]. Another important family of algorithms relies on the approximation of the constraint Jacobian matrices in addition to the Hessian approximation technique, in order to further reduce the computational effort [49, 94, 324]. The SQP subproblem then reads as

$$\min_{\Delta y} \quad \frac{1}{2}\Delta y^\top H \Delta y + a(\bar{y}, \bar{\lambda}, \bar{\nu})^\top \Delta y \tag{1.21a}$$

$$\text{s.t.} \quad 0 \;=\; c(\bar{y}) + A\,\Delta y \qquad | \; \lambda_{\text{QP}} \tag{1.21b}$$

$$0 \;\geq\; h(\bar{y}) + B\,\Delta y \qquad | \; \nu_{\text{QP}}, \tag{1.21c}$$

based on the Jacobian approximations $A \approx c_y(\bar{y})$ and $B \approx h_y(\bar{y})$. These methods typically rely on adjoint derivative computation techniques [141] to efficiently evaluate the modified gradient $a(\bar{y}, \bar{\lambda}, \bar{\nu}) = \nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}, \bar{\nu}) - A^\top \bar{\lambda} - B^\top \bar{\nu}$ in the QP objective (1.21a). This gradient correction is needed in order to allow local convergence to solutions of the original NLP [49, 94, 324]. These algorithms will therefore be referred to as *adjoint based* inexact SQP methods, as discussed further as part of Chapter 7.

Local convergence results

This thesis will focus on local convergence results for equality constrained optimization problems such as those in Theorem 1.26. This is motivated by the following standard result on the local stability of the active set within SQP type methods [49, 52, 232, 284].

Theorem 1.28 (Stability of QP active set) *Consider a local minimizer* $(y^\star, \lambda^\star, \nu^\star)$ *of the NLP* (1.10)*, for which LICQ and strict complementarity*

holds, and the Hessian matrix H in the QP subproblem is positive definite on the nullspace of the linearized active constraints. If a current iterate $(\bar{y}, \bar{\lambda}, \bar{\nu})$ is sufficiently close to the minimizer $(y^\star, \lambda^\star, \nu^\star)$, then the solution of the QP (1.19) *has the same active set as the original NLP.*

The fact that the active set remains stable near a solution allows us to study local convergence properties of SQP methods for a given active set. This explains why exact Hessian schemes locally converge quadratically, Quasi-Newton methods typically result in a superlinear convergence while other techniques such as Gauss-Newton methods converge linearly based on standard Newton-type convergence theory [232]. This is also the reason why our presentation in Chapter 6 and 7 considers equality constrained optimization problems when discussing local Newton-type convergence. Even though Theorem 1.28 is crucial to locally obtain the equivalence of the SQP method with a Newton-type algorithm, the region of full step convergence is typically much larger than the neighborhood where the QP active set is always the same [49].

1.4 Tailored Convex Solvers for Optimal Control

In the previous section, we gave an overview on Nonlinear Programming (NLP) and the popular Newton-type optimization algorithms. When applying an SQP method to the multiple shooting type OCP in Eq. (1.8), each iteration solves the following structured QP subproblem

$$\min_{\Delta X,\, \Delta U} \quad \sum_{i=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}^\top H_i \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix} + g_i^\top \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}$$

$$+ \frac{1}{2} \Delta x_N^\top H_N \Delta x_N + g_N^\top \Delta x_N \tag{1.22a}$$

$$\text{s.t.} \quad 0 = \Delta x_0 - \Delta \hat{x}_0, \tag{1.22b}$$

$$0 = c_i + \frac{\partial \phi(\bar{x}_i, \bar{u}_i)}{\partial (x_i, u_i)} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix} - \Delta x_{i+1}, \qquad i = 0, \ldots, N-1, \tag{1.22c}$$

$$0 \geq h_i + \frac{\partial h(\bar{x}_i, \bar{u}_i)}{\partial (x_i, u_i)} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}, \qquad i = 0, \ldots, N-1, \tag{1.22d}$$

$$0 \geq r + \frac{\partial r(\bar{x}_N)}{\partial x_N} \Delta x_N, \tag{1.22e}$$

where $\bar{X} = [\bar{x}_0^\top, \ldots, \bar{x}_N^\top]^\top$ and $\bar{U} = [\bar{u}_0^\top, \ldots, \bar{u}_{N-1}^\top]^\top$ denote the current state and control trajectory which form the linearization point for the problem functions. In addition, we define the vectors $g_i := \nabla l(\bar{x}_i, \bar{u}_i)$ and $g_N := \nabla m(\bar{x}_N)$, $\Delta\hat{x}_0 = \hat{x}_0 - \bar{x}_0$, $c_i := \phi(\bar{x}_i, \bar{u}_i) - \bar{x}_{i+1}$, $h_i := h(\bar{x}_i, \bar{u}_i)$ and $r := r(\bar{x}_N)$. The terminal cost matrix $H_N := \nabla_x^2 m(\bar{x}_N)$ is defined, while the stage Hessian blocks H_i for $i = 0, \ldots, N-1$ depend on the used approximation technique. The Lagrangian of the NLP in (1.8) is given by

$$\begin{aligned}\mathcal{L}(X, U, \Lambda, \nu) = &\sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) + \lambda_{-1}^\top (x_0 - \hat{x}_0) + \nu_N^\top r(x_N) \\ &+ \sum_{i=0}^{N-1} \lambda_i^\top (\phi(x_i, u_i) - x_{i+1}) + \sum_{i=0}^{N-1} \nu_i^\top h(x_i, u_i),\end{aligned} \tag{1.23}$$

where λ_i for $i = 0, \ldots, N-1$ denote the multipliers corresponding to the continuity constraints (1.8c), λ_{-1} denotes the multiplier of the initial value condition (1.8b) and ν_i for $i = 0, \ldots, N$ denote the multipliers corresponding the inequality constraints in (1.8d) and (1.8e). In case of an exact Hessian SQP method, these block matrices correspond to the following second order derivatives $H_i := \nabla^2_{(x_i, u_i)} \mathcal{L}(\bar{X}, \bar{U}, \bar{\Lambda}, \bar{\nu})$. In case of the common (nonlinear) least squares type objective $l(x_i, u_i) = \frac{1}{2}\|R(x_i, u_i)\|_2^2$, the Gauss-Newton Hessian approximation reads $H_{\mathrm{GN},i} := \nabla R(\bar{x}_i, \bar{u}_i) \nabla R(\bar{x}_i, \bar{u}_i)^\top$.

The QP subproblem in Eq. (1.22) has a particular sparsity structure, such that general-purpose sparse algorithms are typically not recommendable for real-time optimal control applications as discussed in more detail in [121, 123]. There are two common approaches for solving the band structured QP subproblems (1.22) arising in optimal control. The first approach exploits the sparsity structure of the problems by employing tailored direct linear algebra routines within the convex QP solver, while the second eliminates all state deviations via the continuity constraints of Eq. (1.22c) and solves instead a dense QP of significantly smaller dimension. The state elimination is better known as *condensing* as proposed by [51]. As we will show, these approaches can be competitive and they both benefit from rather recent research.

1.4.1 Direct Sparsity Exploitation

As mentioned earlier, convex solvers exist that specifically tackle the sparse block banded structure of the multi-stage QP for direct optimal control. The main advantage of these sparsity exploiting solvers is that their computational complexity can typically scale linearly in the horizon length N due to the use of

tailored linear algebra routines [274, 313]. One popular example is the FORCES code generation framework [97], which exports efficient primal-dual interior point solvers. Note that extensions of these embedded IP techniques can be made to more general convex optimization problems such as quadratically constrained QPs (QCQPs) and second-order cone programs (SOCP) [95]. Another implementation of this structure exploiting IP method for embedded optimization can be found in the HPMPC package [123]. The latter uses improved Riccati recursion based solvers to treat the linear-quadratic (LQ) control problems at each iteration [121]. An efficient implementation of a structure exploiting, primal-barrier IP method can be found in [323].

The dual Newton strategy [117] is another efficient, sparsity exploiting method to solve the multi-stage QP (1.22) as implemented in the qpDUNES software. This algorithm is based on a dual decomposition approach in the equality constraints, in combination with a non-smooth Newton method on the dual problem. The resulting two-level optimization problem is unconstrained in the dual variables and separable in the primal stage variables, resulting in a naturally parallelizable approach. Another important advantage of the dual Newton strategy over interior point methods is that it can benefit from warm-starting in the context of embedded optimization [115].

In addition to the aforementioned Newton-type algorithms, also the family of first-order gradient methods [230] has been shown to provide desirable properties for fast embedded optimization [135, 182, 245, 283]. More specifically, these techniques are typically shown to be relatively easy to code, with a low computational cost per iteration which also scales linearly in the horizon length and with generally good warm-starting capabilities [282]. In addition, for the solution of QPs, these first-order methods have the advantage over general Newton-type algorithms that the required number of iterations to reach a given accuracy can typically be tightly estimated which allows a priori computational complexity certification [230]. Note that this class of algorithms includes the alternating direction method of multipliers (ADMM) [53] and other *operator splitting* methods that have been shown to be quite favorable for fast MPC applications [234, 299]. For example, efficient software implementations can be found in QPgen [135], POGS [243] and FiOrdOs [306]. These splitting techniques have recently even been extended to the non-convex case, tailored to real-time NMPC [172].

1.4.2 Condensing and Expansion

As mentioned earlier, the continuity constraints $\Delta x_{i+1} = c_i + \frac{\partial \phi(\bar{x}_i, \bar{u}_i)}{\partial (x_i, u_i)} \begin{bmatrix} \Delta x_i \\ \Delta u_i \end{bmatrix}$ for $i = 0, \ldots, N-1$ can be used to numerically eliminate the state variables from the QP in the form of a *condensing* procedure [51]. Although the subproblem can then be solved in the reduced variable space consisting of the control inputs Nn_{u} instead of the full space of dimension $(N+1)n_{\mathrm{x}} + Nn_{\mathrm{u}}$, the current state variables in $\bar{X}$ are still updated in each SQP iteration using an *expansion* step [8]. The fact that the NLP iterations are still performed in the full variable space, is the crucial difference with using a single shooting method directly in the space of the control variables. Note that the condensed, small scale QP can be solved by an efficient, dense linear algebra solver such as `qpOASES` which is a parametric active-set method [109].

The computational complexity analysis of condensing has recently been revisited in [19, 25, 122]. These novel results have shown that the corresponding computational cost can be made asymptotically of order $\mathcal{O}(N^2)$, including the factorization cost as discussed in [25, 122]. Employing the condensing technique is known to perform very well for relatively short horizon lengths. But in case of longer horizon lengths N, it can be favorable to use one of the aforementioned structure exploiting solvers which typically have an asymptotically linear complexity $\mathcal{O}(N)$. Comparative simulation results can be found in [313]. Classical condensing is generally a good approach in case of many state variables $n_{\mathrm{x}} > n_{\mathrm{u}}$, while complementary condensing [189] was proposed as a competitive alternative in case of many controls $n_{\mathrm{u}} > n_{\mathrm{x}}$.

1.4.3 Block Condensing: optimal level of sparsity

Whether the sparse or condensed formulation is more appropriate for a certain problem depends mainly on the horizon length and the ratio between number of states and controls [313]. Recently, a hybrid method that tries to combine the advantages of both approaches has been proposed [24]. The idea is based on collecting multiple stages of the QP subproblem, e.g., into groups consisting each of M consecutive stages. For each of these $\tilde{N} = \frac{N}{M}$ groups of stages or *blocks*, one can then eliminate all but the first state variable based on a condensing procedure carried out independently within each block. Note that this approach is naturally parallelizable since the condensing can be performed for each block in parallel. This technique has been referred to as either *partial condensing* [24, 121] or *block condensing* [115, 192] in the literature.

Depending on the structure exploiting algorithm that is used to solve the multi-stage QP (1.22), there is a specific trade-off to be made in the computational complexity analysis related to the dimension of the block variables $n_{\mathrm{x}} + Mn_{\mathrm{u}}$ and the total number of blocks $\tilde{N} = \frac{N}{M}$. By varying the block size M, one can directly impact this trade-off such that an optimal block size can typically be found, resulting in an optimal level of sparsity. A detailed study on this optimal choice of the block size for Riccati based algorithms to solve the linear-quadratic control problems can be found in [24, 121]. It has however been shown that the previously mentioned dual Newton strategy can often profit even more from this block condensing approach, because of the effect that an increased block size has on the number of dual Newton iterations [192].

Let us briefly comment on this combination of the dual Newton strategy, as implemented efficiently in the solver `qpDUNES` [117], together with the block condensing technique as implemented in the `ACADO` code generation tool for real-time NMPC [177] and presented in more detail in [192]. Figure 1.2 was taken as a highlight from [192], which illustrates the computational trade-off when varying the block size M. It can be seen in the upper right part of the figure that the solution of each subproblem, which comprises $n_{\mathrm{x}} + Mn_{\mathrm{u}}$ block variables, naturally becomes more expensive for increasing values of M. At the same time, the number of subproblems $\tilde{N} = \frac{N}{M}$ decreases and typically also the average number of dual Newton iterations as shown in the upper left part of the figure. As a result, the optimal block size can be observed to be about $M = 10$ for the overall computation time in the bottom part of Figure 1.2 on this specific chain mass control example [192]. Because of a similar effect of the block size on the convergence of the outer iterations, this blocking technique has also been shown beneficial for the augmented Lagrangian based alternating direction inexact Newton (ALADIN) method [178] to directly tackle the nonlinear OCP problem in Eq. (1.8) as presented in [194].

1.5 Real-Time Algorithms for MPC and MHE

In the past few sections, we have provided a compact overview on the different algorithmic tools that can be used to implement dynamic optimization based control and estimation techniques. Figure 1.3 repeats the necessary problem transformations to solve the parametric, continuous time OCP in Eq. (1.6). For this purpose, a direct multiple shooting method can be used to parameterize the continuous time problem resulting in a finite dimensional NLP in Eq. (1.8). The nonlinear optimization problem can then be solved using an SQP-type algorithm. This section briefly introduces Model Predictive Control (MPC) and Moving Horizon Estimation (MHE) to respectively compute the next control input or

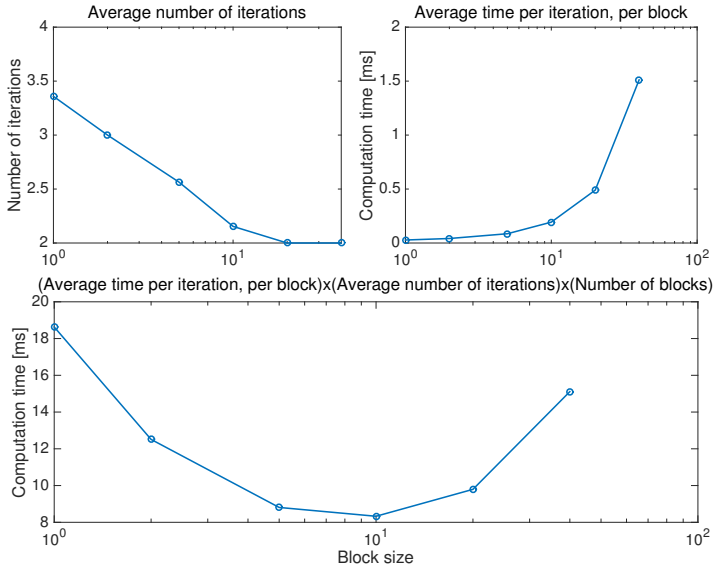


Figure 1.2: Block condensing with `qpDUNES`: trade-off in choosing the block size M for the optimal control of a chain of 4 masses with horizon length $N = 200$.

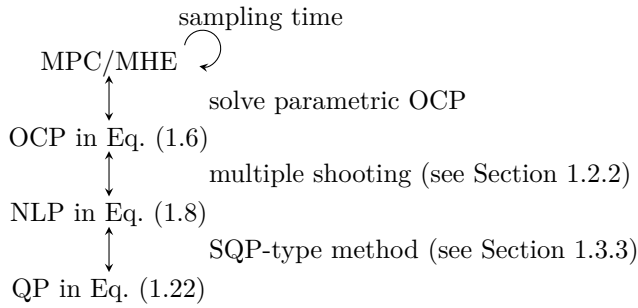


Figure 1.3: Overview of an SQP-type numerical treatment of multiple shooting based real-time optimal control.

state estimate for the system of interest. Both approaches are based on real-time dynamic or *embedded optimization* within a receding horizon framework. We introduce the corresponding online algorithms for optimal control, with a focus on the SQP based real-time iteration (RTI) scheme.

1.5.1 Model Predictive Control

In order to carry out a certain control task, one could formulate and solve a corresponding optimal control problem, e.g., of the form in Eq. (1.6) and simply apply the computed trajectory of control inputs in *open loop* to the system. In an ideal situation, the result of this implementation would be optimal by definition, according to the provided OCP formulation. For any real-world application, one would however have to deal with many uncertainties, such as modeling errors, unforeseen external disturbances and imperfect information regarding the current state of the system. One way to deal with this challenge is by instead using *robust* optimal control formulations which optimize the control trajectory to be applied, under the consideration of the various possible worst-case scenarios. More information on this active topic of research can be found in [31, 173, 312] and references therein. Instead, this thesis focuses on *closing the loop* with feedback control, where one observes the system in order to correct the control actions online [114].

Model Predictive Control (MPC) is an advanced dynamic optimization based strategy for feedback control, that can be used to control a large class of systems [151, 257, 277]. The increasing popularity especially of Nonlinear MPC (NMPC) is due to its ability to directly handle nonlinear dynamics, constraints and objectives which allows one to naturally translate design requirements into mathematical statements. The scheme is based on the solution of a parametric OCP at each sampling instant, for which a quite standard formulation was provided earlier in (1.6). This optimization problem is parametric, since it depends on the current state of the system $\hat{x}_0$ that can be either measured directly or estimated as will be discussed further. The model in Eq. (1.6c) should correspond to an accurate representation of the system dynamics of interest, i.e., it can be based on a set of ODE equations (1.1) or an index-1 DAE system (1.4).

Note that the objective in Eq. (1.6a) consists of the stage cost $\ell(\cdot)$ and the terminal cost $m(\cdot)$, for which the following *tracking* formulation is quite common

$$\int_0^{T_\mathrm{p}} \frac{1}{2}\|R(t, x(t), u(t))\|_2^2 \,\mathrm{d}t + \frac{1}{2}\|R_N(x(T))\|_2^2, \tag{1.24}$$

based on a nonlinear least squares type cost and where T_p will be further referred to as the *prediction horizon* length. As mentioned earlier, such an objective function typically results in computationally more tractable optimization problems because Gauss-Newton based Hessian approximations can be used. This is in contrast with more general *economic* MPC formulations where the stage cost can, for example, represent a minimization of risk or a direct maximization of profit [278]. More specifically, the following tracking

type cost function is quite common for practical formulations of MPC

$$\int_0^{T_\mathrm{p}} \frac{1}{2} \left(\|x(t) - x^{\mathrm{ref}}(t)\|_Q^2 + \|u(t) - u^{\mathrm{ref}}(t)\|_R^2 \right) \mathrm{d}t + \frac{1}{2} \|x(T) - x^{\mathrm{ref}}(T)\|_P^2, \tag{1.25}$$

where the deviation from a specific reference state and control trajectory is directly penalized. The corresponding weighting matrices $Q \in \mathbb{S}_{++}^{n_\mathrm{x}}$, $R \in \mathbb{S}_{++}^{n_\mathrm{u}}$ and $P \in \mathbb{S}_{++}^{n_\mathrm{x}}$ are important tuning parameters in the resulting MPC formulation. For simplicity, these cost matrices are typically chosen to be constant even though they could also be time-varying.

In order to implement the MPC scheme, one additionally needs to decide on a sampling frequency with which the optimal control problems are solved. The sampling time T_s should be short enough with respect to the fast dynamics in the process to be controlled; while it should be long enough to allow the necessary computations to be carried out in time. The resulting receding horizon control strategy can be summarized as follows:

1. get the current state of the system $\hat{x}_0$ from the estimator
2. solve (approximately) the parametric optimization problem in (1.6)
3. apply the optimal control solution $u(t)$, $\forall t \in [0, T_\mathrm{s}]$ to the system
4. repeat step 1-3 after the fixed sampling time of T_s

This concept is additionally illustrated in Figure 1.4 which shows the state and control trajectories at two consecutive sampling time points. The figure demonstrates how the use of the current state estimate introduces feedback into the process and therefore closes the loop. It can be observed in Figure 1.4 that the planned control trajectory and the corresponding predicted states change from one time instant to the next, because of the deviation of the state estimate from the model based prediction. Note that this procedure can be simplified further when the control discretization in Section 1.2.2 is designed such that step 3 corresponds directly to applying the first control input in the discrete time trajectory. For some applications, it can also be useful to implement a *shrinking* horizon version of the above scheme, e.g., in batch operation of chemical engineering processes [257].

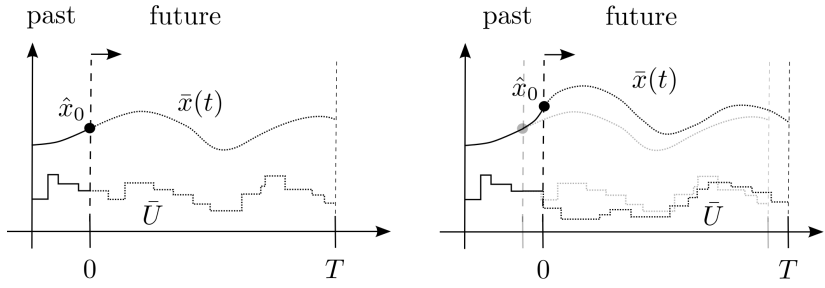


Figure 1.4: Illustration of the receding horizon principle: model predictive control based on the successive solution of optimal control problems.

Because practical implementations of NMPC employ a finite prediction horizon length T_p, it is important to investigate the stability of the resulting feedback control strategy. In addition to the computational attractiveness of a tracking type MPC scheme based on the objectives in either (1.24) or (1.25), these positive definite formulations also considerably simplify the nominal and robust stability analysis of the resulting closed-loop system [17, 151, 277]. Techniques to guarantee stability are typically based on the use of an appropriate terminal ingredient in the form of an (in-)equality constraint and/or a corresponding terminal cost [68]. Other stability results use problem formulations with a sufficiently long control horizon [149] or an additional prediction horizon and a locally stabilizing control law [231]. Recent extensions of such results have also been proposed towards stability of economic MPC formulations [18, 87, 150, 228], even though these topics are outside the scope of this thesis. An alternative approach is based on the design of a tracking cost that locally approximates the economic MPC formulation with the corresponding advantages regarding stability guarantees and the used algorithms, as discussed in [329, 330].

1.5.2 Moving Horizon Estimation

The performance of any feedback control strategy is strongly affected by the accuracy with which the current state of the system can be obtained. In many practical applications, the full state of the system cannot be measured directly such that the design of an observer or state estimator becomes crucial [225]. Note that in addition to the current state of the system, also other system parameters can often be estimated online to aid the control task. One way to tackle this estimation task is by formulating it as a dynamic optimization problem. Moving Horizon Estimation (MHE) considers an *estimation horizon* of fixed length T_e in the past, on which the deviation of the model based

predictions from the actual measurements can be directly minimized [273]. Similar to predictive control, MHE can naturally and optimally take constraints into account. While MPC can be seen as an extension of the Linear Quadratic Regulator (LQR), MHE can be seen as an extension of the Kalman filter for constrained nonlinear systems [276]. In many cases, MHE can outperform an Extended Kalman Filter (EKF) at an increased computational cost [161].

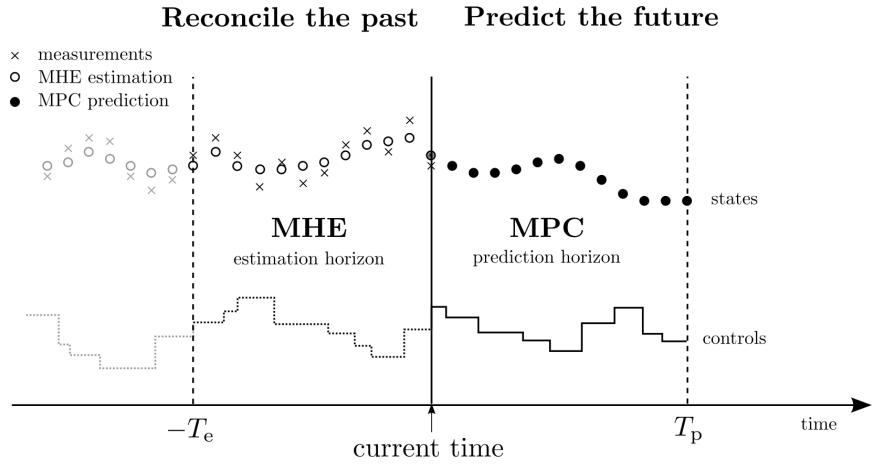


Figure 1.5: Illustration of the estimation horizon T_e in MHE and the prediction horizon T_p in MPC, and the corresponding state and control trajectories at a given time point (inspired by [313]).

The dynamic optimization problem that needs to be solved is nearly a *dual* of the problem in MPC, because it uses the same optimal control structure even though MPC looks into the future while MHE provides an estimate based on measurements in the past. The relation between these two concepts is illustrated in Figure 1.5, showing both the prediction and estimation horizon with respect to a specific time point. The main difference in the optimal control formulation for MHE is that the initial state is free, such that no constraint like Eq. (1.6b) is necessary. The control inputs over the past estimation horizon are already known to the estimator and therefore also do not need to be optimized further. Let us additionally introduce disturbances $w(t)$ to account for a plant-model mismatch, resulting in the following DAE formulation

$$0 = f(\dot{x}(t), x(t), z(t), u(t), w(t)), \tag{1.26}$$

where the control input $u(t)$ can be considered a known, time-varying parameter. Based on this extended dynamic model, the following continuous time MHE

problem formulation could be proposed

$$\min_{x(\cdot),\, w(\cdot)} \quad \int_0^{T_\mathrm{e}} \frac{1}{2}\|R(t, x(t), u(t), w(t))\|_2^2 \,\mathrm{d}t + \frac{1}{2}\|R_0(x(0))\|_2^2 \tag{1.27a}$$

$$\text{s.t.} \quad 0 = f(\dot{x}(t), x(t), z(t), u(t), w(t)), \qquad \forall t \in [0, T_\mathrm{e}], \tag{1.27b}$$

$$0 \geq h(x(t), u(t), w(t)), \qquad \forall t \in [0, T_\mathrm{e}], \tag{1.27c}$$

where the estimate for the current state of the system is provided by $x(T_\mathrm{e})$ at the end of the horizon. Note that practical MHE applications sometimes treat the controls as additional degrees of freedom, using the applied values as measurements in order to account for actuator uncertainty.

Most MHE formulations are based on a convex function to penalize the mismatch between the real measurements and the model based predictions, such as the least squares objective formulation in Eq. (1.27a). In order to increase robustness of the estimates against measurement outliers, the so-called Huber penalty [180] can sometimes be used instead as proposed in [130]. A discussion on how to efficiently deal with this alternative cost function in optimization can be found in [54, 144, 193]. Note that the objective in (1.27a) includes an *arrival cost* term [275], defined by the function $R_0(\cdot)$, which is important for the stability analysis of the resulting MHE scheme [277]. The conceptual idea of this arrival cost is to summarize all prior information that cannot be considered because of the finite length of the estimation horizon. This can be interpreted somewhat similar to the purpose of the terminal cost in MPC, which summarizes the behaviour of the system past the prediction horizon. Practical implementations to update the arrival cost are typically based on variants of an Extended Kalman Filter (EKF), as discussed in [197, 200].

It is important to note that the introduction of path constraints (1.27c) in MHE should be done carefully. Even when the same constraints are imposed in the MPC formulation, it is still possible in practice for the real system to violate them, e.g., due to unknown disturbances. In such a case, one does not want MHE to be negatively biased by these constraints but instead it should correctly estimate these violations. However, such constraints can still be included in the problem formulation, e.g., to avoid unphysical behaviour in the model based predictions. For more information on this and other MHE related topics, the reader is referred to [273, 276]. Note that the disturbances $w(t)$ in the MHE problem (1.27) take a similar role as the control inputs for predictive control. From an algorithmic point of view, we can therefore refer exclusively to the optimization problem in (1.6) to discuss direct optimal control methods and the corresponding online algorithms.

1.5.3 Real-Time Considerations

It would be the dream in nonlinear MPC and MHE to obtain the solution to the next OCP instantly at each sampling point, which is of course impossible because of computational delays. There are, however, online algorithms available to deal with this issue such as described in [13, 183, 190, 211, 235]. A survey and classification of some of these algorithms can be found in [90], on which also this discussion is mainly based. Let us mention some of the common ingredients that are used for real-time optimal control:

- The successive solution of rather similar optimal control problems allows one to effectively initialize the algorithm, using the (approximate) solution at the previous sampling instant. Many continuation techniques are based on sensitivity analysis [61] as will be discussed further.
- It can be beneficial to do as many computations offline as possible and this can go from code optimizations and precomputations to the explicit solution of the control law for small-scale problems [30].
- Instead of solving the MPC problem starting at the current state of the system, it can be a good idea to compensate the computational delay by predicting the state at the time when the problem will be solved [111].
- The necessary computations in each sampling time can be divided into a preparation and a feedback phase [88]. The preparation phase is typically the most CPU intensive one and the feedback phase quickly delivers an approximate solution, once the current state of the system is available.
- Instead of performing iterations for a problem that is getting older and older, it is also possible to work with the most recent information in each new iteration. This idea of postponing computations and allowing the algorithm to *converge over time* is used, e.g., in [88, 212, 235].

More specifically, we are interested in online algorithms for embedded optimization which combine the following key properties:

- **deterministic**: In order to guarantee real-time feasibility, it is important to allow a close to deterministic runtime for the used algorithms. For example, this can rely on a certification of the computational complexity as discussed in [282] or the strict limitation of adaptivity in embedded numerical simulation schemes as discussed in Chapter 2.
- **parallelizability**: Since multi-core platforms become more and more the standard also for embedded control hardware, parallelizability of the

proposed algorithms plays a big role. Even though it is a recurring topic throughout this thesis, extra focus on this aspect is made in Chapter 5.

- **structure exploitation**: Related to the earlier mentioned preparation and optimization of the online computations for embedded optimization, it is important to identify and exploit any common problem structures. One example of this is discussed in Chapter 4 and the concept is directly in line with the principle of automatic code generation in Chapter 8.

Further, we will present the online optimization framework of the Real-Time Iteration (RTI) scheme [89] based on Sequential Quadratic Programming (SQP) which allows one to combine many of the properties above.

1.5.4 Continuation Methods

In Nonlinear MPC, a sequence of optimal control problems with different initial values $\hat{x}_0^{[0]}, \hat{x}_0^{[1]}, \ldots$ needs to be solved. For the transition from one optimization problem to the next, it is beneficial to take into account the fact that the optimal solution depends almost everywhere differentiably on $\hat{x}_0$. Using the concept of parametric NLP sensitivities, one can design a predictor step to obtain a good guess for the solution of the next problem. This is the general idea behind a numerical continuation method as discussed in more detail in [16, 90]. The solution manifold has smooth parts whenever the active set does not change, but non-differentiable points occur where the active set changes [152]. After linearizing at such a point in the context of a nonlinear IP method, a simple *tangential predictor* would lead to a relatively bad approximation. One remedy would be to increase the path parameter τ, which decreases the nonlinearity but it comes at the expense of generally less accurate solutions. Such a tangential predictor is, for example, used in the C/GMRES method presented in [235], the advanced step NMPC controller [333] and a more recent decomposition based algorithm for distributed NMPC [172].

One can deal with these active set changes more naturally in an SQP type framework by the following procedure proposed and analyzed in [88, 304, 332]. First, the parameter $\hat{x}_0$ needs to enter the NLP linearly which is automatically the case for a simultaneous OCP formulation such as in Eq. (1.8). The optimization problem needs to be addressed using an exact Hessian based SQP method [232]. Finally, the solution trajectories for the current problem in $\hat{x}_0^{[k]}$ are used as initial guess to formulate and solve the SQP subproblem for the new parameter value $\hat{x}_0^{[k+1]}$. Note that *shift initialization* strategies are typically used to account for the horizon moving forward in time [38, 93, 212]. The resulting *generalized* tangential predictor can work across active set changes [90] and is

also referred to as *initial value embedding* [88]. This alternative continuation strategy is used in the RTI scheme [89] as well as the multi-level iterations [49], which are presented next. Note that in practical NMPC applications, an approximate variant of this tangential predictor can be used, e.g., based on a Gauss-Newton Hessian approximation.

1.5.5 Real-Time Iterations

Based on the previous discussion of online algorithms for Nonlinear MPC and corresponding continuation methods to exploit the similarity between subsequent dynamic optimization problems, we finally introduce the RTI scheme as presented in [88, 89]. The method is based on SQP (see Section 1.3.3) to iteratively solve the direct multiple shooting OCP in Eq. (1.8). Since it is an online algorithm that avoids performing iterations for a problem that is only getting older, each iteration attempts to use new information from the system. This results in one SQP type iteration per sampling time for the RTI scheme, including either new measurements in the case of MHE or a new state estimate in the case of MPC. The standard implementation is based on the Generalized Gauss-Newton (GGN) method for least squares type objectives [48], as presented in Algorithm 1. Note that extensions of this approach have been proposed, including Jacobian approximations [49] or even second order sensitivities [268] (see Chapter 3). In the latter case, it is important to include sparsity preserving regularization techniques as described in [310].

Algorithm 1 Real-Time Iteration (RTI) scheme using Gauss-Newton

Input: initial guess $X^{[0]} = (x_0^{[0]}, \ldots, x_N^{[0]})$, $U^{[0]} = (u_0^{[0]}, \ldots, u_{N-1}^{[0]})$, $k = 0$

while true **do**

`Preparation phase`

1. Call an integrator to obtain the results $\phi(x_i^{[k]}, u_i^{[k]})$ and their first order sensitivities $\frac{\partial \phi(x_i^{[k]}, u_i^{[k]})}{\partial (x_i, u_i)}$ for $i = 0, \ldots, N-1$ (see Chapter 2).
2. Linearize the path constraints as in (1.22) and evaluate the Gauss-Newton Hessian blocks $H_{\mathrm{GN},i} = \nabla R(x_i^{[k]}, u_i^{[k]}) \nabla R(x_i^{[k]}, u_i^{[k]})^\top$.
3. *Optional*: one can apply condensing to eliminate the state variables and prepare the dense QP as described in Section 1.4.2.
4. Wait until the current state estimate $\hat{x}_0$ arrives.

`Feedback phase`

5. Solve the structured QP subproblem in (1.22), apply the full SQP step $X^{[k+1]} = X^{[k]} + \Delta X$, $U^{[k+1]} = U^{[k]} + \Delta U$ and send the new control input $u_0^{[k+1]}$ to the process.
6. Increment k and shift the trajectories X and U forward in time.

end while

The first important property to observe in Algorithm 1 is that the computations in each iteration are divided into a preparation and a feedback phase [88]. The typically more CPU intensive *preparation phase* is performed with a predicted state, before the current state estimate is even available. This part of the algorithm is naturally parallelizable, because each linearization for $i = 0, \ldots, N-1$ in steps 1-2 can be done independently and therefore in parallel. Once the value $\hat{x}_0$ becomes available, the *feedback phase* quickly delivers a solution guess to the new problem by solving the prepared, convex SQP subproblem. This idea is in line with the earlier mentioned concept of using the most current information in each iteration, i.e., to allow the algorithm to converge over time. Note that the solution of the QP subproblem in step 5 employs the initial value embedding strategy. By performing the problem linearization in steps 1 and 2 using the previous *shifted* solution guess, but including the new state estimate in the initial value condition, a combined predictor-corrector step is obtained at the cost of only one QP solution [90]. This algorithmic technique arises naturally here by using the multiple shooting problem formulation in Eq. (1.8), where the parameter value $\hat{x}_0$ enters linearly.

It has been shown in [91] that, under some reasonable assumptions, the nominal stability of the closed-loop system using the RTI scheme in Algorithm 1 can be guaranteed also in presence of inaccuracies, model errors or disturbances. And even though the RTI scheme was originally proposed as an efficient algorithm for Nonlinear MPC [88], it has been shown to be quite competitive also for nonlinear MHE including corresponding convergence guarantees [328]. It is thereby important to include a numerically efficient arrival cost update scheme within the MHE algorithm such as, e.g., based on a QR factorization as proposed in [200]. Examples of real-world applications using MHE in combination with nonlinear MPC, based on the concept of performing real-time iterations independently for both problem formulations, have been presented in [145, 198, 315]. This idea is illustrated in Figure 1.6, where the interactions within the closed-loop system between the MHE estimator, the MPC controller and the real process are depicted. In this figure, $\hat{x}_0$ denotes the current state estimate, u_0 is the control input applied to the system and y_e denotes the latest measurement. Efficient implementations exist, based on the concept of automatic code generation [222, 237], of the RTI scheme for fast nonlinear MPC [177] as well as fast MHE [107]. More information can also be found in Chapter 8 on the open-source `ACADO` code generation tool.

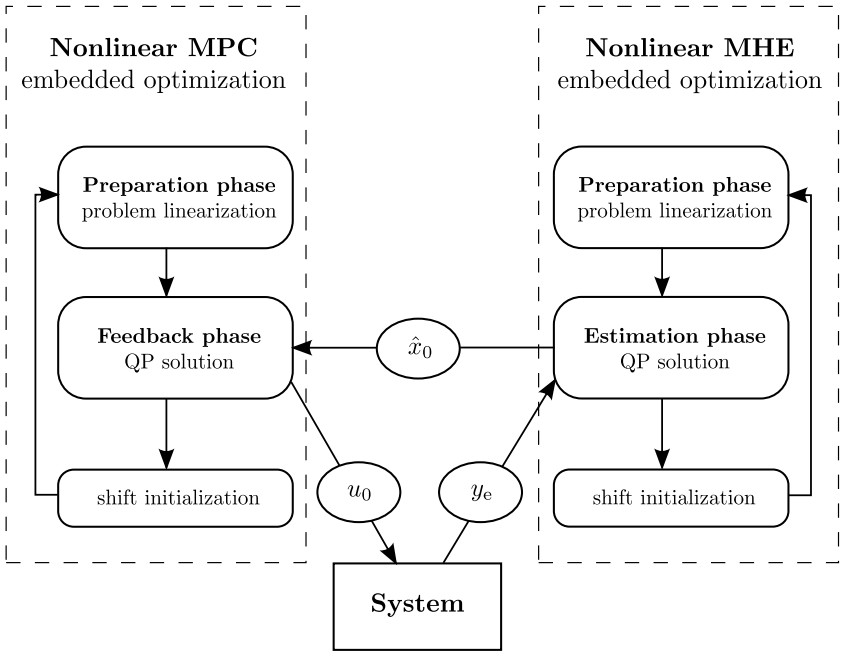


Figure 1.6: Illustration of the RTI scheme for nonlinear MPC and MHE, including their interactions within the closed-loop system (inspired by [115]).

The specific scheduling in time of the computations within the RTI framework is illustrated in Figure 1.7. The feedback phase in the context of MHE is thereby also referred to as the estimation phase. It can be observed in this figure how the RTI scheme allows one to minimize the computational delay between obtaining a new measurement y_e from the system and applying the next control input u_0. An interesting extension of this concept is known as the multi-level or mixed-level iterations, as presented in [49, 118]. This online algorithm typically consists of 4 different levels of QP updates, which we refer to as level A, B, C and D. An iteration on the highest level (D) corresponds to the standard RTI scheme since it computes a Newton-type step based on a complete problem linearization. The other three levels attempt to reduce the overall computational burden, or alternatively to increase the sampling frequency, by reusing specific parts of the QP data. At the lowest level (A), one solves the same QP but only with the new state estimate $\hat{x}_0$ which corresponds to a linear MPC scheme that at least takes active set changes into account [90]. The intermediate levels are typically performed less frequently and are based on partial updates of the QP data, excluding the matrices such that, e.g., factorizations can be reused. A feasibility improvement in level (B) is based on

the evaluation of the nonlinear constraints, while the optimality improvement in level (C) computes the Lagrange gradient in the form of the adjoint based SQP [324] method in Eq. (1.21). Note that a full SQP step on the highest level (D) needs to be performed whenever the system linearization does not allow Newton-type contraction as discussed in [49].

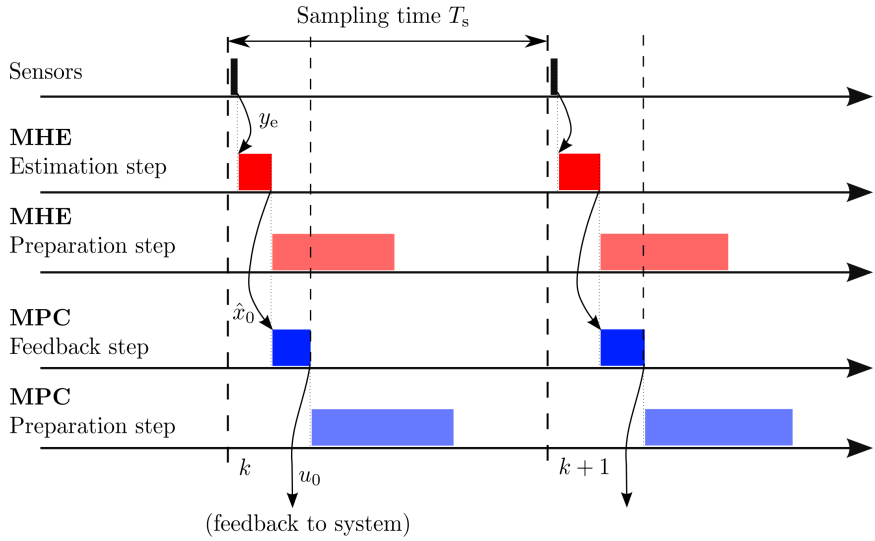


Figure 1.7: Illustration of the scheduling in time of the computations within the RTI framework for embedded optimization (inspired by [313]).

Chapter 2

Numerical Simulation and Sensitivity Propagation

Shooting based optimal control methods rely on an accurate discretization of the original continuous-time OCP, resulting in a finite dimensional NLP [36, 48]. In the context of NMPC, these OCPs have to be solved under strict timing constraints [90]. In many practical control applications, the numerical simulation of the nonlinear dynamics as well as the propagation of first or even higher order derivative information is the main bottleneck in terms of computation time. The current chapter provides an overview on numerical integration schemes and on the task of performing a corresponding sensitivity analysis. A more detailed discussion will be presented for the collocation methods, which form a subclass of Implicit Runge-Kutta (IRK) schemes, and their implementation within an embedded optimization algorithm.

This chapter includes the continuous output based optimal control formulations as presented originally in [261] and [271]. In addition, it presents the embedded integrators based on the article in [269] and the master thesis in [259].

Outline The chapter is organized as follows. Section 2.1 first presents a few of the important families of numerical integration schemes, with a focus on single-step methods. The class of Implicit Runge-Kutta (IRK) schemes is discussed in Section 2.2 as a popular approach to simulate both stiff or implicit systems of differential equations. Section 2.3 then provides a detailed discussion on the tailored propagation of sensitivities for numerical integration schemes, in the context of direct optimal control methods. The implementation of the

specific class of collocation methods within embedded optimization algorithms is discussed in Section 2.4, including efficient sensitivity propagation schemes. Finally, Section 2.5 presents and motivates the use of a continuous output feature within the optimal control problem formulation.

2.1 Numerical Integration Methods

Section 1.1.4 gave already an introduction into the domain of the numerical simulation of differential equations, e.g., including the concepts of local and global error propagation and convergence of the integration scheme. This section aims at providing a compact overview on the numerical solution of ordinary differential equations (ODE), with a further discussion on the treatment of algebraic equations in the next section. For this purpose, let us consider the following explicit version of the IVP from Eq. (1.3)

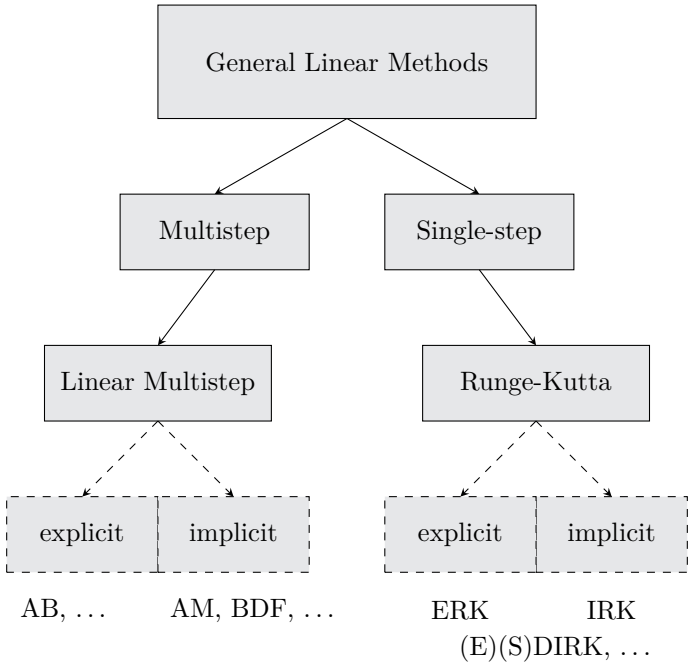
$$\dot{x}(t) = f_{\mathrm{e}}(t, x(t)), \quad x(t_0) = \hat{x}_0, \tag{2.1}$$

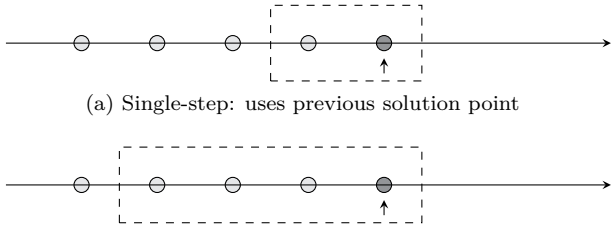
where the initial value $\hat{x}_0$ is given. Note that there are no control inputs $u(t)$ included in this IVP, because they are assumed to be known from the perspective of the numerical simulation routine. The existence and uniqueness of the solution is guaranteed under the assumptions in Theorem 1.2.

2.1.1 General Linear Methods

Each numerical integration method takes a step forward in time, e.g., over the interval $t \in [t_n, t_{n+1}]$ in order to find an approximate value for the solution point $x(t_{n+1})$. A first distinguishing feature is then whether this numerical method employs merely the previous point $x_n \approx x(t_n)$ or a set of previously computed solution values $x_n, x_{n-1}, \ldots$ (and their function values). The first approach is typically referred to as *single-step* methods, while the latter is called a *multistep* method as illustrated also in Figure 2.2. These two families are collectively referred to as general linear methods, based on a linear combination of solution points and function evaluations [63]. A compact and therefore incomplete overview of these numerical integration schemes is provided in Figure 2.1. Note that most single-step methods use additional, auxiliary function evaluations in order to obtain a certain order of accuracy (see Definition 1.8). Multistep methods, on the other hand, provide their order of accuracy by storing more solution points and they therefore require a good starting procedure [157].

Figure 2.1: An incomplete overview of numerical integration schemes.





(a) Single-step: uses previous solution point

(b) Multistep: uses certain amount of previous points

Figure 2.2: Illustration of single-step versus multistep integration methods.

In this thesis, the focus will be on single-step methods. They do not need a starting procedure, which makes them rather suitable for short simulation times as they are needed in simultaneous approaches for direct optimal control (see Section 1.2). Especially for embedded optimization where a relatively low accuracy is required, a few integration steps of a high order Runge-Kutta (RK)

method is typically sufficient. Similar advantages of single-step schemes over multistep methods for the solution of dynamic optimization problems have been observed by [199, 291]. Nevertheless, we briefly mention some of the linear multistep (LM) methods for completeness.

Linear multistep methods

An s-step Linear Multistep (LM) method uses the solution points x_i and the corresponding function evaluations $f_i = f_{\mathrm{e}}(t_i, x_i)$, for $i = n - s + 1, \ldots, n$ to compute the next solution value $x_{n+1} \approx x(t_{n+1})$

$$x_{n+1} + \alpha_{s-1} x_n \; + \; \ldots \; + \; \alpha_0 x_{n-s+1} \; = \\ T_{\mathrm{int}} \left(\beta_s f_{n+1} \; + \; \beta_{s-1} f_n \; + \; \ldots \; + \; \beta_0 f_{n-s+1} \right), \tag{2.2}$$

where T_{int} refers to the integration step size and α_i, β_i are the constants defining the method, with at least α_0 or β_0 different from zero. The scheme can be written more compactly as

$$x_{n+1} = T_{\mathrm{int}} \sum_{i=1}^{s+1} \beta_{i-1} f_{n-s+i} - \sum_{i=1}^{s} \alpha_{i-1} x_{n-s+i}, \tag{2.3}$$

which defines the value x_{n+1} explicitly in the case $\beta_s = 0$ and implicitly when $\beta_s \neq 0$. In the latter case of an implicit LM method, a procedure is needed in order to solve the resulting nonlinear equation (2.3).

One popular family of LM schemes is known as the *Adams methods*, on which more information can be found in [157]. These numerical methods are constructed by integrating a polynomial interpolation as an approximation of the system right-hand side in (2.1). When including the new value (t_{n+1}, f_{n+1}) in the interpolation points, an implicit Adams method can be obtained. These explicit and implicit formulas are also respectively referred to as Adams-Bashforth (AB) and Adams-Moulton (AM) methods [39]. Another important class of LM methods consists of the Backward Differentiation Formulas (BDF). Unlike the Adams methods, BDF formulas are based on numerical differentiation. The BDF methods of order up to 5 are quite popular also for direct optimal control [7, 27, 166], because of their good stability properties for stiff differential equations as discussed in the next section.

Remark 2.1 *The main distinction in our presentation of integration methods will be between stiff and nonstiff problems, considering general differential equations. There are however other important classes of problems with a special geometric structure, such as Hamiltonian dynamics [207]. For this purpose,*

different types of structure-preserving or geometric integrators have been proposed and studied extensively [72, 156, 205]. An interesting application example in the optimal control for mechatronic systems is based on the $SO(3)$ orthogonal Lie group, typically used to represent the orientation of a body in a 3-dimensional space [184, 297]. As discussed further in [28, 146], an alternative to using tailored integrators to preserve the corresponding invariants is to perform a Baumgarte stabilization directly into the system dynamics in order to avoid numerical drift. It is important to note that these effects occur more prominently for long simulation times and therefore will not be our main concern in embedded optimization for direct optimal control, where the horizon length is relatively short and the required accuracy is rather low.

2.1.2 Single-Step Methods

Unlike LM formulas, single-step methods refer to only one previous solution point and its function evaluation to determine the current value. The family of Runge-Kutta (RK) methods is the most important class of single-step schemes, which are generically applicable to ODEs. Unlike LM methods based on a linear combination of previous solution points as in (2.3), RK methods perform additional function evaluations at intermediate stage points in order to achieve a high order of accuracy. Let us introduce the following general formulation of an RK method with q stages

$$\begin{aligned} k_1 &= f_{\mathrm{e}}(t_n + c_1 T_{\mathrm{int}}, x_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{1j} k_j), \\ &\vdots \\ k_q &= f_{\mathrm{e}}(t_n + c_q T_{\mathrm{int}}, x_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{qj} k_j), \\ x_{n+1} &= x_n + T_{\mathrm{int}} \sum_{i=1}^{q} b_i k_i, \end{aligned} \tag{2.4}$$

in which b_i denote the weights and a_{ij} the internal coefficients of the method for $i = 1, \ldots, q$ and $j = 1, \ldots, q$. These expressions are sometimes referred to as the *differential* formulation, since $k_i \in \mathbb{R}^{n_{\mathrm{x}}}$ for $i = 1, \ldots, q$ denote the state derivatives at the stages points [39]. An alternative formulation of the same RK method, based directly on the stage values $X_i \in \mathbb{R}^{n_{\mathrm{x}}}$ for the differential states,

can be written as

$$\begin{aligned} X_1 &= x_n + T_{\text{int}} \sum_{j=1}^{q} a_{1j} f_{\text{e}}(t_n + c_j T_{\text{int}}, X_j), \\ &\vdots \\ X_q &= x_n + T_{\text{int}} \sum_{j=1}^{q} a_{qj} f_{\text{e}}(t_n + c_j T_{\text{int}}, X_j), \\ x_{n+1} &= x_n + T_{\text{int}} \sum_{i=1}^{q} b_i f_{\text{e}}(t_n + c_i T_{\text{int}}, X_i), \end{aligned} \tag{2.5}$$

which is then referred to as the *integral* formulation. It is mathematically equivalent to Eq. (2.4). In numerical experiments, one can however experience a different conditioning for both formulations, especially in the case of implicit methods as shown in [23].

The internal coefficients a_{ij} satisfy the following consistency condition

$$c_i = \sum_{j=1}^{q} a_{ij}, \tag{2.6}$$

which simplifies the derivation of order conditions [157]. For the method to have at least order 1, the values b_i must satisfy

$$\sum_{i=1}^{q} b_i = 1, \tag{2.7}$$

which is the first of the Runge-Kutta order conditions. The formulas in either Eq. (2.4) or (2.5) are typically represented uniquely using a Butcher [63] tableau such as found below.

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1q} \\ \vdots & \vdots & & \vdots \\ c_q & a_{q1} & \cdots & a_{qq} \\ \hline & b_1 & \cdots & b_q \end{array} \tag{2.8}$$

2.1.3 Explicit Runge-Kutta Schemes

Explicit RK (ERK) methods have the property that the stage values k_i in (2.4) or X_i in (2.5) can be obtained explicitly, and this successively for $i = 1, \ldots, q$.

This means that the coefficient matrix $A = (a_{ij})$ is strictly lower triangular in the Butcher tableau (2.8). The simplest RK method is the forward Euler method, $x_{n+1} = x_n + T_{\text{int}} f_{\text{e}}(t_n, x_n)$, which is only of order 1. A more practical explicit RK method is the following 4-stage scheme of order 4

$$\begin{array}{c|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \tag{2.9}$$

This RK method is so commonly used that it is often referred to as the *classical* Runge-Kutta method of order 4 (RK4). ERK methods of order up to 8 are among the most widely used schemes for nonstiff problems [157].

2.1.4 Implicit Runge-Kutta Schemes

Implicit RK (IRK) methods are described by a matrix $A = (a_{ij})$ that is not strictly lower triangular, but rather fully dense in general. This means that the expressions in (2.4) or (2.5) correspond to a system of $n_{\text{x}} \times q$ nonlinear equations, which need to be solved in order to obtain the stage values. For this purpose, Newton-type methods are typically used, as detailed further in Section 2.4. It is therefore clear that the implementation of implicit RK methods typically requires an increased computational effort compared to an explicit variant. However, IRK methods often have a higher order of accuracy for the same amount of stages q and they exhibit better stability properties in case of stiff differential equations [157, 158].

The simplest example of an implicit RK method is the backward or implicit Euler method, described by $x_{n+1} = x_n + T_{\text{int}} f_{\text{e}}(t_n + T_{\text{int}}, x_{n+1})$. Even though this scheme is still only of order 1, it has superior stability properties over its explicit forward variant. Another well-known, implicit RK method is the trapezoidal rule which is defined by the following Butcher tableau:

$$x_{n+1} = x_n + \frac{T_{\text{int}}}{2}(f_{\text{e}}(t_n, x_n) + f_{\text{e}}(t_{n+1}, x_{n+1}))$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array} \tag{2.10}$$

Note that the trapezoidal rule is *stiffly accurate* [14], which means that the last stage value corresponds to the solution at the end of the integration step.

More specifically for a q-stage method, this means that $c_q = 1$ and $a_{qi} = b_i$ for $i = 1, \ldots, q$. This property will be important in the discussion on stiff problems and systems with algebraic equations [256]. Further examples of implicit RK methods will follow in the next section.

2.1.5 Semi-Implicit Runge-Kutta

When the coefficient matrix of the RK method is neither strictly lower triangular nor fully dense but instead exhibits a particular structure which has computational advantages, then such a method is often called *semi-implicit* [223]. In the case of fully implicit RK methods, a large nonlinear system of $n_{\mathrm{x}} \times q$ equations (2.4) needs to be solved. For Diagonally Implicit RK (DIRK) methods, on the other hand, the sequential computation of each stage value corresponds to a separate nonlinear system that needs to be solved. The coefficient matrix A is then lower triangular, as illustrated in Figure 2.3. When DIRK methods have only identical diagonal elements in the matrix A (all equal to γ in Figure 2.3), they are called Singly DIRK (SDIRK) methods. For the nonlinear system corresponding each stage, the same Jacobian approximation and its factorization can then be reused in the Newton-type iterations. As a special case, an explicit SDIRK (ESDIRK) method additionally has an explicit first stage. Even though such methods can significantly lower the computational cost, they can often still offer good stability properties as discussed in [199] for a specific 4-stage ESDIRK method. For an overview on semi-implicit RK methods, see [14, 64, 158, 233].

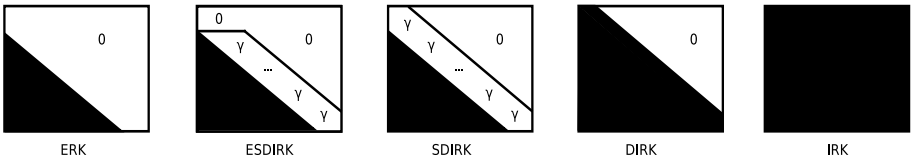


Figure 2.3: Structure of the matrix A for different families of RK methods [199].

2.2 Implicit Runge-Kutta Methods

In addition to their desirable properties as discussed in this section, it is important to note that implicit integration methods can handle implicit systems of differential equations as easily as the explicit ODE formulation in (2.1). Let us therefore consider the following implicitly defined IVP

$$0 = f(t, \dot{x}(t), x(t)), \quad x(t_0) = \hat{x}_0. \tag{2.11}$$

In addition, this section includes a discussion on the treatment of index-1 DAE systems by IRK schemes

$$\begin{aligned} 0 &= f(t, \dot{x}(t), x(t), z(t)), \quad x(t_0) = \hat{x}_0, \\ 0 &= g(t, x(t), z(t)), \end{aligned} \tag{2.12}$$

which corresponds to the semi-explicit formulation in (1.5) where the algebraic variables $z(t)$ are defined by a separate set of algebraic equations $g(\cdot) = 0$ for notational convenience. These methods can however also be applied directly to the fully implicit DAE formulation from Eq. (1.4).

2.2.1 Stiff Differential Equations

Even though they are not as straightforward to implement as explicit schemes, implicit integration methods are quite popular because of their superior performance on stiff problems which are rather common in practice. Many references would even describe stiff differential equations as those for which certain implicit methods perform considerably better than explicit ones [158]. As we will discuss next, the eigenvalues of the Jacobian of the system equations certainly play a role but also many other quantities such as the system dimension and smoothness of the solution are important. We introduce one typical characterization of stiffness, followed by a discussion on desirable stability properties for numerical simulation.

Characterization of stiffness

An important class of stiff problems consists of processes with multiple time scales, i.e., including fast as well as relatively slow dynamics. For this purpose, we introduce a simple characterization of stiffness based on the following linearized dynamics at a certain time point

$$\dot{x}(t) = A(t)x(t) + \phi(t), \tag{2.13}$$

where A is a $n_\mathrm{x} \times n_\mathrm{x}$ matrix with n_x eigenvalues λ_i and we assume that $\mathrm{Re}(\lambda_i) < 0$ holds for $i = 1, \ldots, n_\mathrm{x}$. A relatively small value of $|\mathrm{Re}(\lambda_i)|$ then corresponds to slow dynamics, for which it would be desirable to take rather large integration steps. On the other hand, a larger value of $|\mathrm{Re}(\lambda_i)|$ typically requires a smaller integration step size T_int for numerical stability. The following concept of a *stiffness ratio* [203] can therefore be used

$$\frac{\max\limits_{i=1,\ldots,n_\mathrm{x}} |\mathrm{Re}(\lambda_i)|}{\min\limits_{i=1,\ldots,n_\mathrm{x}} |\mathrm{Re}(\lambda_i)|}, \tag{2.14}$$

which becomes larger for systems that are more likely to be stiff. Note that this ratio is generally time-dependent for a nonlinear system, such that stiffness needs to be defined within a certain domain. As mentioned earlier and discussed in more detail by [63, 158, 203], this definition does not cover all possible cases of stiff differential equations.

Numerical stability properties

Let us apply an RK method as in Eq. (2.4) to the linear test equation $\dot{x}(t) = \lambda x(t)$, which is often referred to as *Dahlquist*'s equation [75]. After solving the resulting expressions for the next solution point x_{n+1}, an explicit function can be obtained in terms of the previous point x_n

$$x_{n+1} = R(T_{\text{int}}\lambda)\, x_n = R(z)\, x_n, \tag{2.15}$$

where the shorthand notation $z = T_{\text{int}}\lambda$ is used. This function $R(\cdot)$ is typically referred to as the stability function of the numerical integration method. Using Eq. (2.15), it should be clear that the relation between the solution point x_n and the initial value x_0 can be written as $x_n = R(z)^n x_0$. The region of absolute stability is defined as

$$\{z \in \mathbb{C} |\ |R(z)| < 1\}. \tag{2.16}$$

For the values of z within this region, it holds that the numerical solution to the linear test problem decays to zero $x_n \to 0$ as $n \to \infty$. A large region of absolute stability is therefore a desirable property for a method.

More specifically, a numerical integration method whose region of absolute stability contains the left complex half-plane $\mathbb{C}^-$ is called *A-stable*. Note that also the exact solution of Dahlquist's equation $x(t) = x_0 e^{\lambda t}$ is stable if $\lambda \in \mathbb{C}^-$. When applying an A-stable integration method to a stiff system of differential equations, the step size T_{int} is generally not limited anymore because of stability issues. Another desirable property that needs to be mentioned is stiff A-stability or L-stability [14, 256], which is stronger than A-stability. If $|R(z)| \to 1$ for $z \to -\infty$, then it is possible that the stiff components of the numerical method are damped out rather slowly compared to the exact solution. A method is called *L-stable* if it is A-stable and if the stability function satisfies $|R(z)| \to 0$ for $z \to -\infty$. It is interesting to note that an A-stable RK method with nonsingular coefficient matrix A and which additionally is stiffly accurate, can be shown to be L-stable [158]. Even though the mentioned stability definitions will be sufficient for our purposes throughout this thesis, alternative concepts can further be found in [14, 63, 158, 203, 256].

2.2.2 Collocation Methods

Let us now introduce an important family of IRK schemes, the *collocation methods*, which are known to have good stability properties. For ordinary differential equations, the idea corresponds to defining a polynomial $p(t)$ of degree q whose derivative coincides at q given nodes with the vector field of the differential equation [157]. For this purpose, q distinct collocation points $0 \leq c_1 < c_2 < \ldots < c_q \leq 1$ need to be provided which uniquely define the corresponding integration method. This is also illustrated in Figure 2.4 for one specific integration step. The collocation polynomial $p(t)$ needs to satisfy the following $q+1$ conditions:

$$\begin{aligned} p(t_n) &= x_n \\ 0 &= f(t_i, \dot{p}(t_i), p(t_i)) \quad \text{for} \quad i = 1, \ldots, q, \end{aligned} \tag{2.17}$$

using the implicit ODE formulation in (2.11) and based on the time points $t_i := t_n + c_i T_{\text{int}}$ for $i = 1, \ldots, q$.

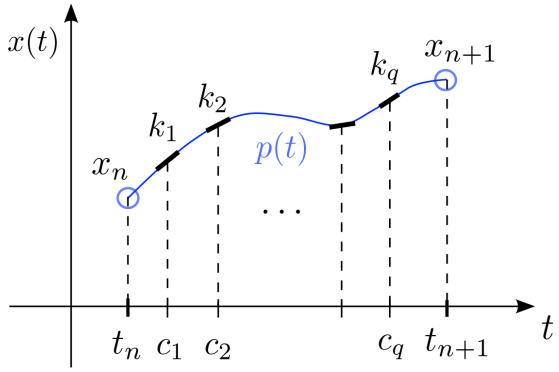


Figure 2.4: Illustration of one integration step of a collocation method based on a polynomial interpolation through a set of points c_i, $i = 1, \ldots, q$.

Based on this definition of the collocation polynomial, a numerical approximation for the solution at time $t_{n+1} = t_n + T_{\text{int}}$ can be obtained simply by evaluating the polynomial, i.e., $x_{n+1} = p(t_n + T_{\text{int}})$. The resulting integration method can be shown to be equivalent to an implicit RK method [157, 158], where the values in the Butcher tableau (2.8) read as

$$a_{ij} = \int_0^{c_i} \ell_j(\tau)\,\mathrm{d}\tau, \quad b_i = \int_0^{1} \ell_i(\tau)\,\mathrm{d}\tau, \quad i, j = 1, \ldots, q, \tag{2.18}$$

where $\ell_i(t)$ denote the Lagrange interpolating polynomials

$$\ell_i(t) = \prod_{\substack{j=1 \\ j \neq i}}^{q} \frac{t - c_j}{c_i - c_j}, \quad i = 1, \ldots, q. \tag{2.19}$$

Even though all collocation methods are part of the IRK family, not all IRK methods can be interpreted as collocation schemes. For example, the coefficients c_i need to be distinct for a collocation method which is also referred to as a *nonconfluent* RK method [203]. It is interesting to note that collocation methods offer a continuous approximation of the solution for the system of differential equations on the complete interval $[t_n, t_{n+1}]$, which is a useful feature also for optimal control as discussed further in Section 2.5.

For an overview of collocation methods and their numerical properties, the reader is referred to [157, 158]. We introduce two popular classes of collocation schemes: the *Gauss* and *Radau IIA* methods.

The Gauss-Legendre collocation methods

From the previous discussion, it is clear that only the quadrature nodes c_i are needed to define a collocation method. We first introduce the Gauss-Legendre schemes, for which the q nodes are placed at the roots of a shifted Legendre polynomial as illustrated in Figure 2.5. The methods corresponding to $q = 1$, 2 and 3 (order $P = 2$, 4 and 6) are often used and they have as collocation points

$$c_1 = \frac{1}{2}, \qquad q = 1, \quad P = 2, \tag{2.20}$$

$$c_1 = \frac{1}{2} - \frac{\sqrt{3}}{6},\ c_2 = \frac{1}{2} + \frac{\sqrt{3}}{6}, \qquad q = 2, \quad P = 4, \tag{2.21}$$

$$c_1 = \frac{1}{2} - \frac{\sqrt{15}}{10},\ c_2 = \frac{1}{2},\ c_3 = \frac{1}{2} + \frac{\sqrt{15}}{10}, \quad q = 3, \quad P = 6. \tag{2.22}$$

The complete Butcher tableaus, corresponding to the expressions in (2.18), can be found in [157, 158]. The Gauss methods are quite popular because they can be shown to be optimal in terms of the order of accuracy [158], i.e., they provide the highest possible order of accuracy $P = 2\,q$ for a specific number of stages. In addition, the Gauss-Legendre collocation schemes are A-stable which is important for stiff systems of equations. Note that the order result refers to the end point order P, while the internal stages of any collocation method have an order of at least q [39].

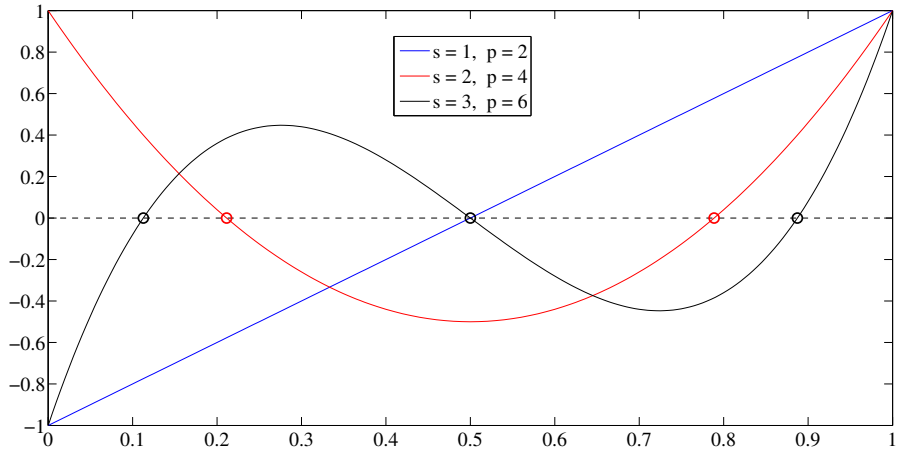


Figure 2.5: The roots of the shifted Legendre polynomials of order 1, 2 and 3, which are respectively used as nodes for the Gauss methods of order 2, 4 and 6.

The Radau IIA collocation methods

The Gauss methods are attractive because of their high order of accuracy, even though this class of collocation schemes can be shown not to be L-stable. The alternative is to use a q-stage Radau IIA method which has order $P = 2\,q - 1$ (one order lower than the corresponding Gauss method), but they are L-stable and therefore provide a desirable damping property for very stiff problems. These methods are based on choosing the zeros of particular polynomials as the collocation nodes

$$c_1 = 1, \qquad q = 1, \quad P = 1, \tag{2.23}$$

$$c_1 = \frac{1}{3},\ c_2 = 1, \qquad q = 2, \quad P = 3, \tag{2.24}$$

$$c_1 = \frac{4 - \sqrt{6}}{10},\ c_2 = \frac{4 + \sqrt{6}}{10},\ c_3 = 1, \quad q = 3, \quad P = 5. \tag{2.25}$$

In addition to their improved stability properties, it can be observed that Radau IIA methods always include the end point $c_q = 1$ in the collocation nodes. This property was previously referred to as being stiffly accurate [14], which is desirable for very stiff systems and for differential-algebraic equations as discussed further. These two classes will be used throughout this thesis,

while other interesting families of A-stable methods can be found such as the various Lobatto collocation formulas [158].

2.2.3 Alternative Methods for Stiff Problems

We further focus mainly on collocation methods for direct optimal control, but it is important to mention some of the other popular integration schemes especially for stiff systems of differential equations. A first important family consists of the earlier mentioned Backward Differentiation Formulas (BDF). Even though the BDF methods of order up to 5 are quite popular in practice for stiff problems, none of the linear multistep methods of order larger than 2 is A-stable. Instead, the higher order BDF schemes are A(α)-stable which is a weaker but still practical condition as specified in [158].

BDF and collocation methods are quite popular for stiff problems, but they are both fully implicit and therefore require an implementation based on a Newton-type method. Alternative schemes are based on a single linear system solution in order to retain certain stability properties, which are often referred to as *linearly implicit* methods. One simple example is the linearly implicit Euler method, for which an integration step reads as $(\mathbb{1}-T_{\text{int}}\frac{\partial f_n}{\partial x})(x_{n+1}-x_n) = T_{\text{int}} f_n$, when $f_n = f_{\text{e}}(x_n)$ for an explicit ODE (2.1). Also higher order methods can be constructed such as, for example, the family of *Rosenbrock* methods [158, 159]. Because of their computational advantages, linearly implicit integration schemes are quite popular also for real-time applications [21].

A third and final family of alternative integration schemes for stiff problems, which should be mentioned here, is based on the exponential of the Jacobian matrix. These schemes are known as *exponential* integration methods [168, 169]. They recently gained an increased interest because of efficient iterative methods to compute vector-products of a matrix exponential based, for example, on Krylov subspace projection methods [158]. The general idea of an exponential integrator is to integrate the linear part of the continuous-time system exactly, which often helps to mitigate the stiffness of the differential equations. Note that this class of integration methods forms an interesting connection between linear or linear time varying (LTV) MPC schemes and online algorithms for nonlinear MPC as discussed in [147].

2.2.4 Differential-Algebraic Equations

Most integration methods can be extended to DAE systems of index 1 in a rather straightforward manner, since the algebraic variables can in principle

be eliminated numerically (see Definition 1.3). More specifically, the implicit function theorem states that a locally unique solution $z = G(x)$ exists such that the index-1 DAE in (2.12) can be rewritten as an implicit ODE

$$0 = f(t, \dot{x}(t), x(t), G(x(t))), \quad x(t_0) = \hat{x}_0. \tag{2.26}$$

Because these variables $z(t)$ are generally defined implicitly by the algebraic equations, explicit DAE solvers are typically not very competitive [22, 294] and implicit integration schemes are therefore preferred. For the numerical analysis using the DAE formulation in (2.12), it is interesting to note that it corresponds to a singular perturbation problem

$$\begin{aligned} 0 &= f(t, \dot{x}(t), x(t), z(t)), \quad x(t_0) = \hat{x}_0, \\ \epsilon\, \dot{z}(t) &= g(t, x(t), z(t)), \end{aligned} \tag{2.27}$$

for the limit case of the parameter $\epsilon \to 0$, as discussed in [158]. The numerical simulation of an index-1 DAE by reformulating it into an ODE system as in (2.26) is often referred to as an *indirect* approach. We are however interested in a *direct* treatment of differential-algebraic equations using implicit RK methods, such that the result is equivalent to the indirect approach.

Similar to Eq. (2.4) for an explicit ODE system, the differential formulation of an IRK method applied to the DAE system (2.12) reads as

$$0 = f(t_n + c_i T_{\text{int}}, k_i, x_n + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i), \quad i = 1, \ldots, q, \tag{2.28a}$$

$$0 = g(t_n + c_i T_{\text{int}}, x_n + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i), \qquad i = 1, \ldots, q, \tag{2.28b}$$

$$x_{n+1} = x_n + T_{\text{int}} \sum_{i=1}^{q} b_i k_i, \tag{2.28c}$$

$$0 = g(t_{n+1}, x_{n+1}, z_{n+1}), \tag{2.28d}$$

where $t_{n+1} = t_n + T_{\text{int}}$ and the additional variables $Z_i \in \mathbb{R}^{n_z}$ for $i = 1, \ldots, q$ denote the stage values for the algebraic variables. Note that also an integral formulation of the IRK method could be used, similar to Eq. (2.5) for the explicit ODE system. The expression in Eq. (2.28b) forces the stage values Z_i to be consistent at each of the collocation nodes $i = 1, \ldots, q$, as defined directly by the algebraic equations in an index-1 system. In a similar fashion, Eq. (2.28d) is needed in order to obtain a fully consistent solution point (x_{n+1}, z_{n+1}) at

the end of the integration step. Note that the structure due to the algebraic stage variables can be exploited when solving the nonlinear system consisting of Eqs. (2.28a) and (2.28b), as discussed in more detail in [259].

Stiffly accurate IRK schemes form a particularly interesting class of numerical simulation methods for DAE systems since their last stage corresponds to the end of the integration step, i.e., $c_q = 1$ and $z_{n+1} = Z_q$ such that Eq. (2.28d) is automatically satisfied because of (2.28b). As mentioned earlier, an A-stable and stiffly accurate IRK method with invertible coefficient matrix A can be shown to be L-stable. This turns out to be an important property both for very stiff problems and for differential-algebraic equations, as discussed in more detail in [56, 158]. A popular family of IRK methods for DAE systems therefore consists of the Radau IIA collocation methods.

Remark 2.2 *Many interesting processes are described naturally by higher index DAE systems. One can rely on index reduction techniques [39, 241] to reformulate such a set of equations into an index-*1 *DAE system with associated consistency conditions. The alternative is the direct solution of these higher index DAEs, using specialized numerical methods as discussed in more detail in [22, 56, 158]. Note that also stiffly accurate collocation methods can be extended to higher index DAE systems [158].*

2.3 Efficient Sensitivity Propagation

This section introduces different approaches to compute directional sensitivities of the simulation result with respect to the initial conditions. More specifically, let $x_T(x_0, u)$ denote the numerical approximation of the solution for the explicit (2.1) or implicit (2.12) IVP over the interval $[0, T]$. Using Newton-type optimization for direct optimal control as discussed in Sections 1.2-1.4, one typically needs the following first order derivatives

$$\frac{\mathrm{d}x_T(x_0, u)}{\mathrm{d}(x_0, u)} \in \mathbb{R}^{n_\mathrm{x} \times (n_\mathrm{x} + n_\mathrm{u})}, \tag{2.29}$$

where $x_0 = \hat{x}_0$ denotes the initial state value and u are the control inputs applied to the dynamic system. For simplicity, the control values are assumed constant over the integration interval $[0, T]$ which corresponds to a piecewise constant parameterization in direct optimal control. In addition, a dynamic optimization formulation as in Eq. (1.6) is considered such that the sensitivities of the algebraic variables are not generally needed even though we will comment on how they can be computed easily. In the case of adjoint based Newton-type optimization techniques such as in Eq. (1.21), one is additionally interested in

the adjoint first order sensitivities

$$\frac{\mathrm{d}x_T(x_0,u)}{\mathrm{d}(x_0,u)}^\top \lambda \in \mathbb{R}^{(n_\mathrm{x}+n_\mathrm{u})}, \tag{2.30}$$

where $\lambda \in \mathbb{R}^{n_\mathrm{x}}$ denotes the backward seed. For a detailed discussion on the propagation of second order sensitivities, the reader is referred to the next chapter. This is needed for the computation of the Hessian of the Lagrangian in Newton-type optimization. Note that also other applications require the efficient and accurate computation of sensitivities, e.g., in the analysis of dynamic systems or in model reduction strategies [204].

2.3.1 Algorithmic Differentiation

We start by briefly introducing the concept of Algorithmic Differentiation (AD), which allows the efficient evaluation of first and higher order derivatives of factorable, computer represented functions. Directional derivatives can be computed up to machine precision and this at a computational cost that is of the same order of magnitude as the cost of evaluating the original, differentiable function. More details on these concepts and efficient implementations can be found in [6, 19, 40, 141]. These AD techniques can be used further within an implementation of sensitivity propagation for numerical integration schemes.

Each differentiable function consists of several elementary functions and operations such as multiplication, addition, trigonometric functions etc. The evaluation of such a function can be represented as an expression graph or algorithm description. The principle of AD is then to apply the chain rule and differentiate each of the elementary operations separately in order to obtain a new algorithm to evaluate the derivatives of the original function. There are generally two different ways to implement AD. The first approach is based on computing the derivatives during or right after the evaluation of the non-differentiated function and is typically referred to as *operator overloading*. The name of this technique originates from the fact that it is mostly implemented using operator overloading in advanced programming languages such as C++. Efficient implementations can be found, for example, in `ADOL-C` [142] and `CppAD` [2]. The alternative is to represent the differentiated algorithm as a separate expression graph that can be evaluated. This approach is known as *source code transformation*. Efficient implementations of the latter technique can be found in `ADIC` [1] and `CasADi` [20].

Independent of the specific AD implementation, the expression graph for the differentiated function can be traversed either in the same order as the original function evaluation or in the reverse order. These two options are referred to

as respectively the *forward* and *reverse* or backward mode of AD to compute first order derivatives. Let us consider a specific function $f : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ of input dimension n_1 and output dimension n_2. The forward mode of AD to compute a directional derivative is then much cheaper than the reverse mode in case $n_1 \ll n_2$. Note that the forward mode of AD is slightly more expensive than using numerical finite differences, but it is exact up to machine precision, while finite differences generally lead to a numerical loss of about half the valid digits [6]. Conversely, the backward mode of AD can be much faster than both finite differences and the forward mode of AD in case the function has much less outputs than inputs, i.e., $n_2 \ll n_1$.

The main disadvantage of the backward mode of AD is that it typically has a much higher memory footprint than forward schemes, because of the intermediate variables and partial derivatives which need to be stored. Practical implementations of reverse AD therefore often rely on a certain trade-off between the computational cost and memory requirements, by using *checkpointing* techniques as described in [141, 319]. It allows one to store only the intermediate variables at certain checkpoints during the forward evaluation. A more detailed analysis of the implementation and computational complexity for these AD techniques can be found in [40, 141]. Note that the distinction between forward and backward techniques becomes especially important when combining these approaches in order to compute higher order derivatives [141, 240], as discussed also in the next chapter.

2.3.2 Sensitivities for Direct Optimal Control

We focus in the following sections on the required first order sensitivities within direct methods for dynamic optimization. A discussion on higher order differentiation techniques can be found in [6] or in the next chapter. The model functions in the IVP are assumed to be sufficiently smooth, in accordance with Assumption 1.10. There are generally two classes of sensitivity propagation techniques considered, which differ from each other depending on whether they are based on a *differentiate-then-discretize* or a *discretize-then-differentiate* type of approach. Before we detail how either of these approaches could be applied together with the aforementioned numerical simulation methods, let us discuss some of their advantages and disadvantages.

The differentiate-then-discretize approach is based on a continuous-time sensitivity propagation. The original system of differential equations is extended with the appropriate set of linear variational equations, depending on which directional derivatives are required. The main advantage of this approach is that it naturally allows a sensitivity analysis for the exact solution of the original

continuous-time dynamic system. The resulting system of variational equations can be discretized with an arbitrary accuracy, using any numerical integration scheme. When using derivative-based optimization for direct optimal control (see Section 1.2), one is however interested in accurate derivative information for the approximately discretized representation of the state trajectory solution. In embedded optimization especially where the discretization accuracy is often rather low, it is still important to accurately compute the corresponding sensitivities in order to obtain convergence for the Newton-type algorithm.

Because of the above observation, a discrete-time sensitivity propagation following the discretize-then-differentiate principle is typically preferred for direct optimal control as well as for embedded optimization applications. This class of techniques will therefore be the main focus throughout this thesis. As discussed in more detail further, a discrete-time propagation of the sensitivities can sometimes be shown to be equivalent to using the exact same integration scheme for the numerical simulation of the sensitivity equations. But even in such a case where both approaches become mathematically equivalent, a discretize-then-differentiate technique automatically results in a tailored exploitation of the structure in the linear variational equations and therefore can often be computationally more efficient [6].

2.3.3 Differentiate-then-Discretize

By applying forward differentiation directly to the IVP, one can obtain a new system of equations for the corresponding forward sensitivities as discussed in [67, 106]. Let us do this for Eq. (1.2) based on an explicit ODE formulation, by differentiating with respect to both the initial state value x_0 and the constant control inputs u which results in

$$\begin{aligned} \dot{S}_x(t) &= \frac{\partial f_{\mathrm{e}}}{\partial x}(x(t), u(t)) S_x(t), & S_x(0) &= \mathbb{1}, \\ \dot{S}_u(t) &= \frac{\partial f_{\mathrm{e}}}{\partial x}(x(t), u(t)) S_u(t) + \frac{\partial f_{\mathrm{e}}}{\partial u}(x(t), u(t)), & S_u(0) &= \mathbb{0}, \end{aligned} \tag{2.31}$$

where the additional state derivative variables $S_x(t) := \frac{\mathrm{d}x(t)}{\mathrm{d}x_0} \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{x}}}$ and $S_u(t) := \frac{\mathrm{d}x(t)}{\mathrm{d}u} \in \mathbb{R}^{n_{\mathrm{x}} \times n_{\mathrm{u}}}$ are defined. Eq. (2.31) corresponds to a fully determined IVP, since the initial values read as $S_x(0) = \frac{\mathrm{d}x(0)}{\mathrm{d}x_0} = \mathbb{1}$ and $S_u(0) = \frac{\mathrm{d}x(0)}{\mathrm{d}u} = \mathbb{0}$. This set of sensitivity equations would typically be referred to as the system of Variational Differential Equations (VDE). By forward simulation over the interval $[0, T]$, one obtains directly the full Jacobian $\frac{\mathrm{d}x(T)}{\mathrm{d}(x_0, u)} = \begin{bmatrix} S_x(T) & S_u(T) \end{bmatrix}$. Note that we can construct this VDE system also for

a specific set of n_s forward directions $\frac{\mathrm{d}x(T)}{\mathrm{d}(x_0,u)}\bar{S}$, where the matrix $\bar{S} \in \mathbb{R}^{(n_\mathrm{x}+n_\mathrm{u})\times n_\mathrm{s}}$ would be referred to as the forward seed.

This technique can be applied in a similar fashion to implicit systems and to DAEs, where the resulting sensitivity equations would typically be referred to as the Variational Differential-Algebraic Equations (VDAE). Let us do this once for the fully implicit DAE formulation of index-1 from Definition 1.3, by differentiating with respect to the control inputs resulting in

$$0 = \frac{\partial f}{\partial \dot{x}}(t)\dot{S}^x(t) + \frac{\partial f}{\partial x}(t)S^x(t) + \frac{\partial f}{\partial z}(t)S^z(t) + \frac{\partial f}{\partial u}(t), \quad S^x(0) = \mathbb{0}, \tag{2.32}$$

where the additional differential $S^x(t) := \frac{\mathrm{d}x(t)}{\mathrm{d}u} \in \mathbb{R}^{n_\mathrm{x}\times n_\mathrm{u}}$ and algebraic $S^z(t) := \frac{\mathrm{d}z(t)}{\mathrm{d}u} \in \mathbb{R}^{n_\mathrm{z}\times n_\mathrm{u}}$ state derivative variables are defined. Note that both versions of forward sensitivity equations are linear and they clearly have the same stiffness properties as the original IVP. Furthermore, the VDAE system in (2.32) is of index 1 as a direct corollary of Definition 1.3.

Adjoint sensitivity propagation

Inspired by the earlier discussion on the forward and reverse mode of Algorithmic Differentiation (AD), it can sometimes also be more efficient to propagate the continuous-time sensitivity results backward in time. The resulting linear variational equations are typically referred to as the *adjoint* system, for which a detailed derivation can be found in [65, 66]. Instead of repeating these results here, we briefly present the adjoint variant of the forward system of variational equations (2.31). For this purpose, we need to introduce a backward seed $\bar{\lambda} \in \mathbb{R}^{n_\mathrm{x}}$ in order to define the directional derivative $\frac{\mathrm{d}x(T)}{\mathrm{d}(x_0,u)}^\top \bar{\lambda}$. The adjoint system for the explicit ODE formulation in (1.2), for which the VDE system was written in (2.31), then reads as follows

$$\begin{aligned} -\dot{\lambda}_x(t) &= \frac{\partial f_\mathrm{e}}{\partial x}(x(t),u(t))^\top \lambda_x(t), \qquad \lambda_x(T) = \bar{\lambda}, \\ -\dot{\lambda}_u(t) &= \frac{\partial f_\mathrm{e}}{\partial u}(x(t),u(t))^\top \lambda_x(t), \qquad \lambda_u(T) = \mathbb{0}, \end{aligned} \tag{2.33}$$

where the additional state derivative variables $\lambda_x(t) := \frac{\mathrm{d}x(T)}{\mathrm{d}x(t)}^\top \bar{\lambda} \in \mathbb{R}^{n_\mathrm{x}}$ and $\lambda_u(t) \in \mathbb{R}^{n_\mathrm{u}}$ are defined. This system of equations denotes an IVP that needs to be solved backward in time, given the values $\lambda_x(T) = \frac{\mathrm{d}x(T)}{\mathrm{d}x(T)}^\top \bar{\lambda} = \bar{\lambda}$ and $\lambda_u(T) = \mathbb{0}$ at the end of the interval. The adjoint simulation result then corresponds to the directional derivative $\frac{\mathrm{d}x(T)}{\mathrm{d}(x_0,u)}^\top \bar{\lambda} = \begin{bmatrix}\lambda_x(0)\\ \lambda_u(0)\end{bmatrix}$.

The adjoint system for the fully implicit index-1 DAE equations can similarly be found in [65], including a discussion on the augmented reformulation which is often used for reasons of stability in [66]. Note that adjoint sensitivity analysis can be more efficient than the standard forward propagation, whenever the derivatives with respect to many inputs are needed. For example, in the case where the full Jacobian matrix $\frac{\mathrm{d}x(T)}{\mathrm{d}(x_0,u)}$ needs to be evaluated and $(n_\mathrm{x}+n_\mathrm{u}) \gg n_\mathrm{x}$, such that a lot of forward sensitivity directions would be needed. The main drawback of this alternative approach is that the original IVP and the adjoint variational system cannot be solved simultaneously, but the simulation results of the nominal system are generally needed for the solution of the adjoint IVP. The resulting implementation consists typically of a forward-backward propagation where the original system is simulated first using a forward sweep, in which most of the intermediate results need to be stored in order to perform a subsequent backward sweep to simulate the adjoint system in (2.33).

Implementation details

For the forward simulation of the system of sensitivity equations in (2.31) or (2.32), one could provide the extended dynamic system together with the equations from the original IVP to a general-purpose numerical integration code in a black-box manner. This would however generally not be computationally efficient because the structure in the linear variational equations would not be exploited in that case. More specifically, the close relation between the original and the variational IVP allows one to reuse many variables throughout the solution process. For this reason, tailored solver implementations have been developed such as in the software `DASSL/DASPK` [56, 220] and as part of the `SUNDIALS` package [166]. Note that a discretize-then-differentiate type of approach, as discussed further in more detail, automatically exploits this structure in the variational equations by construction.

The presented adjoint sensitivity propagation techniques rely on the ability to simulate a certain system of differential equations both forward and backward in time. Otherwise, one could employ a time reversing transformation as discussed in [65]. An efficient implementation of the resulting forward-backward propagation for first order derivatives can be found in [166], which employs checkpointing in order to reduce the storage requirements for an adjoint sensitivity analysis. There is the additional complication that the discretization grids for the original IVP and the adjoint system in general do not coincide, such that the necessary values in the backward propagation are possibly not computed in the prior forward sweep. A typical practical solution to this problem is based on an interpolation, using the stored solution values on the discretization grid of the forward propagation [56]. This again increases the

computational effort and possibly introduces extra numerical errors, compared to a tailored discrete-time sensitivity propagation.

2.3.4 Discretize-then-Differentiate

In order to perform the differentiation task after discretization, resulting in a discrete-time sensitivity propagation, one needs a way to differentiate the integration scheme itself. For this purpose, there are two rather different types of approaches. The first and most classical technique was already mentioned earlier as an alternative to AD in order to evaluate derivatives of a factorable function. One can consider the integrator as a black box and compute its derivatives, just like any other black-box function that can be called externally, by using finite differences (FD). When $x_T(x_0, u)$ again denotes the numerical simulation result for the IVP, let us, e.g., compute the derivative with respect to one of the control inputs $\frac{\mathrm{d}x_T(x_0,u)}{\mathrm{d}u}\bar{u} \in \mathbb{R}^{n_\mathrm{x}}$ where the seed $\bar{u} \in \mathbb{R}^{n_\mathrm{u}}$ contains once the value 1 and zeros everywhere else. Using finite differences, this sensitivity result can be numerically approximated as

$$\frac{\mathrm{d}x_T(x_0,u)}{\mathrm{d}u}\bar{u} \approx \frac{x_T(x_0,u+\delta\bar{u}) - x_T(x_0,u)}{\delta}, \tag{2.34}$$

where δ is the quantity of the perturbation of the input in the direction $\bar{u}$. Because the same integrator is simply called multiple times with different values for the input, this technique is typically referred to as External Numerical Differentiation (END). The standard rule of thumb for FD applies here that, in case the value δ is optimally chosen, about half of the valid digits are typically lost compared to the numerical accuracy of the function evaluation. In the case of an adaptive integrator with error control, there is however the additional problem that each call with different inputs might lead to a different discretization grid. Small values for δ can therefore lead to discontinuous perturbations, e.g., due to a change in the step size and order selection strategy. For this reason, the perturbation should be chosen large enough, e.g., $\delta = \sqrt{\mathrm{TOL}}$ in case TOL denotes the integrator accuracy and the FD result preserves about half the valid digits with respect to this accuracy [6]. Note that this technique requires $(n_\mathrm{u} + 1)$ times the cost of a forward simulation in order to compute the Jacobian $\frac{\mathrm{d}x_T(x_0,u)}{\mathrm{d}u}$. Even though the obtained numerical derivatives are relatively inaccurate, the END scheme is easy to implement.

The alternative idea, which instead is typically referred to as Internal Numerical Differentiation (IND) [43, 48], is to freeze the adaptive components of the integrator such as, e.g., the step size, order, iteration matrices and number of Newton-type iterations, and to differentiate the resulting discretization scheme instead of the adaptive code itself. The fixed integrator function can then

be differentiated using any technique, including finite differences (FD) and algorithmic differentiation (AD). One thereby sometimes distinguishes between IND based on FD, versus the term internal algorithmic differentiation when using AD. We will collectively refer to these techniques as IND, but rather make a distinction based on whether the iterative procedures in an implicit integration scheme are differentiated or instead a direct approach is used. These two options will respectively be denoted as either an *iterative* or *direct* IND scheme. Any type of IND approach to sensitivity analysis generally requires a more involved implementation of an integrator with tailored sensitivity propagation, but it has the computational advantage that a lot of intermediate values from the nominal simulation can be reused. As mentioned earlier, the IND scheme also provides the exact derivatives of the numerical simulation result directly, which is important for Newton-type optimization. In what follows, we discuss more detailed the application of IND to the family of RK methods including both explicit (ERK) and implicit (IRK) schemes.

Remark 2.3 *Even though we considered adaptive integration schemes with error control in the above discussion of external versus internal differentiation techniques, it was mentioned already in the previous chapter that often fixed step integrators are used for embedded applications because of the resulting deterministic runtime and ease of parallelization. It has been observed here that there is an additional advantage for the implementation of the sensitivity analysis when fixing the step size and order, because the resulting integrator then depends continuously on its inputs. The importance of adaptivity for embedded optimization will however be discussed in Section 2.4.*

2.3.5 Sensitivity Propagation for ERK Methods

The implementation of an explicit integration scheme does not include any iterative procedure such that a relatively easy differentiable mapping is obtained after fixing the step size and order for sensitivity analysis. Let us recall the RK formulation from Eq. (2.4) and rewrite the explicit variant in a more compact way as follows

$$\begin{aligned} k_i &= f_{\mathrm{e}}(t_n + c_i T_{\mathrm{int}}, x_n + T_{\mathrm{int}} \sum_{j=1}^{i-1} a_{ij} k_j), \quad i = 1, \ldots, q, \\ x_{n+1} &= x_n + T_{\mathrm{int}} \sum_{i=1}^{q} b_i k_i. \end{aligned} \tag{2.35}$$

When applying forward differentiation directly to these explicit expressions, one obtains a new numerical integration scheme to propagate the corresponding

sensitivities. It has been observed in [15, 287, 320] that the forward differentiated equations are exactly equivalent to those obtained when applying the same explicit method directly to the forward VDE system (2.31). We illustrate this principle, by differentiating the ERK method in (2.35) with respect to the initial value x_0, resulting in

$$\begin{aligned} \frac{\mathrm{d}k_i}{\mathrm{d}x_0} &= \frac{\partial f_\mathrm{e}}{\partial x_n}(t_n + c_i T_\mathrm{int}) \left(S_n + T_\mathrm{int} \sum_{j=1}^{i-1} a_{ij} \frac{\mathrm{d}k_j}{\mathrm{d}x_0} \right), \quad i = 1, \ldots, q, \\ S_{n+1} &= S_n + T_\mathrm{int} \sum_{i=1}^{q} b_i \frac{\mathrm{d}k_i}{\mathrm{d}x_0}, \end{aligned} \tag{2.36}$$

where the additional variables $S_n = \frac{\mathrm{d}x_n}{\mathrm{d}x_0}$ have been introduced, for which the initial value $S_0 = \mathbb{1}$. These expressions indeed denote the same ERK formula, but applied to the differential equation for the sensitivity matrix $S_x(t)$ in (2.31). Note that the partial derivatives of the dynamics in (2.36) could still be evaluated with finite differences, even though we further prefer to use AD for its numerical accuracy as well as its computational efficiency.

Also adjoint sensitivity results for an ERK method can be obtained directly by applying the reverse mode of AD to the integration scheme, after fixing the discretization grid following the IND principle. As mentioned earlier, such an adjoint sensitivity propagation scheme can be more efficient than the forward method, whenever the state derivatives with respect to relatively many inputs are needed. The main disadvantage of using an adjoint propagation technique is the typically high corresponding memory requirement, since the intermediate values in the forward simulation need to be stored. Alternatively, a checkpointing technique could be applied [319]. Note that the connection between an AD based approach and the adjoint system of variational equations in (2.33) has been studied in [154, 155, 320]. As part of their results, they state mild symmetry conditions that need to be satisfied by an ERK method such that the reverse mode of AD results in an integration scheme for the adjoint system of sensitivity equations with the same convergence properties.

Adjoint sensitivity propagation: the RK4 method

We look at an interesting example that is considered also in [320], featuring the 4-stage RK method from Eq. (2.9) which is of order 4. Let us write this scheme

in the standard ERK formulation (2.35) as follows

$$\begin{aligned} k_1 &= f_{\mathrm{e}}(t_n, x_n), \\ k_2 &= f_{\mathrm{e}}(t_n + \frac{T_{\mathrm{int}}}{2}, x_n + \frac{T_{\mathrm{int}}}{2} k_1), \\ k_3 &= f_{\mathrm{e}}(t_n + \frac{T_{\mathrm{int}}}{2}, x_n + \frac{T_{\mathrm{int}}}{2} k_2), \\ k_4 &= f_{\mathrm{e}}(t_n + T_{\mathrm{int}}, x_n + T_{\mathrm{int}} k_3), \\ x_{n+1} &= x_n + \frac{T_{\mathrm{int}}}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{2.37}$$

By applying the reverse mode of AD to these expressions, in order to propagate the adjoint variables $\lambda_n = \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_n}^\top \lambda_{n+1}$, with respect to the states, one obtains the following equations

$$\begin{aligned} l_1 &= \frac{\partial f_{\mathrm{e}}^4}{\partial x}^\top \frac{T_{\mathrm{int}}}{6} \lambda_{n+1} = \frac{T_{\mathrm{int}}}{6} \frac{\partial f_{\mathrm{e}}^4}{\partial x}^\top \lambda_{n+1}, \\ l_2 &= \frac{\partial f_{\mathrm{e}}^3}{\partial x}^\top \left(2\frac{T_{\mathrm{int}}}{6} \lambda_{n+1} + T_{\mathrm{int}}\, l_1 \right) = 2\frac{T_{\mathrm{int}}}{6} \frac{\partial f_{\mathrm{e}}^3}{\partial x}^\top \left(\lambda_{n+1} + \frac{6}{2}\, l_1 \right), \\ l_3 &= \frac{\partial f_{\mathrm{e}}^2}{\partial x}^\top \left(2\frac{T_{\mathrm{int}}}{6} \lambda_{n+1} + \frac{T_{\mathrm{int}}}{2}\, l_2 \right) = 2\frac{T_{\mathrm{int}}}{6} \frac{\partial f_{\mathrm{e}}^2}{\partial x}^\top \left(\lambda_{n+1} + \frac{6}{4}\, l_2 \right), \\ l_4 &= \frac{\partial f_{\mathrm{e}}^1}{\partial x}^\top \left(\frac{T_{\mathrm{int}}}{6} \lambda_{n+1} + \frac{T_{\mathrm{int}}}{2}\, l_3 \right) = \frac{T_{\mathrm{int}}}{6} \frac{\partial f_{\mathrm{e}}^1}{\partial x}^\top \left(\lambda_{n+1} + \frac{6}{2}\, l_3 \right), \\ \lambda_n &= \lambda_{n+1} + l_1 + l_2 + l_3 + l_4, \end{aligned} \tag{2.38}$$

where we used the compact notation $f_{\mathrm{e}}^i := f_{\mathrm{e}}(t_n + c_i T_{\mathrm{int}}, x_n + T_{\mathrm{int}} \sum_{j=1}^{i-1} a_{ij} k_j)$ to denote the function evaluations for $i = 1, \ldots, 4$. Let us define auxiliary

variables $\tilde{k}_i = \frac{l_i}{T_{\text{int}} b_i}$, such that the expressions can be rewritten as

$$\begin{aligned}
\tilde{k}_1 &= \frac{\partial f_{\text{e}}^4}{\partial x}^\top \lambda_{n+1}, \\
\tilde{k}_2 &= \frac{\partial f_{\text{e}}^3}{\partial x}^\top \left(\lambda_{n+1} + \frac{T_{\text{int}}}{2}\,\tilde{k}_1\right), \\
\tilde{k}_3 &= \frac{\partial f_{\text{e}}^2}{\partial x}^\top \left(\lambda_{n+1} + \frac{T_{\text{int}}}{2}\,\tilde{k}_2\right), \\
\tilde{k}_4 &= \frac{\partial f_{\text{e}}^1}{\partial x}^\top \left(\lambda_{n+1} + T_{\text{int}}\,\tilde{k}_3\right), \\
\lambda_n &= \lambda_{n+1} + \frac{T_{\text{int}}}{6}\left(\tilde{k}_1 + 2\tilde{k}_2 + 2\tilde{k}_3 + \tilde{k}_4\right),
\end{aligned} \tag{2.39}$$

which corresponds to the same 4-stage ERK scheme applied backwards to the adjoint system $-\dot{\lambda}(t) = \frac{\partial f_{\text{e}}}{\partial x}(t, x(t))^\top \lambda(t)$, where the final value is given by the backward seed $\lambda_T = \bar{\lambda}$. Note that the stored values from the forward simulation and the corresponding function evaluations and their derivatives are used in the reversed order $i = 4, \ldots, 1$, because of the backward in time propagation of the sensitivities.

2.3.6 Sensitivity Propagation for IRK Methods

It becomes even more important to exploit the structure that is present in the system of variational equations, such as in Eq. (2.31) or (2.32), when using an implicit integration scheme. The reason for this is the relatively high computational cost of the linear algebra routines which are typically needed in the standard implementation of an IRK method such as in Eq. (2.28). A discrete-time sensitivity propagation approach would however automatically exploit such a structure and it additionally allows the reuse of many intermediate values, as will be discussed in Section 2.4. Regarding a discretize-then-differentiate based sensitivity analysis for IRK methods, there are two alternative approaches as discussed also in [6]. A strict application of the IND principle to the implementation of an implicit integrator leads to a differentiation of the iterative procedure to solve the nonlinear equations and will therefore be referred to as an iterative IND scheme. One can alternatively assume that the implicit equations are solved exactly, even if that is not precisely the case in practice, and derive a direct expression for the corresponding sensitivities by applying the implicit function theorem (IFT).

Direct differentiation approach

Similar to the case in (2.36) for ERK methods, one can directly apply forward differentiation to the fully implicit RK formula from Eq. (2.4) or Eq. (2.28) for DAEs. Let us perform this once with respect to the initial value x_0 for the general IRK formulation in (2.28), resulting in

$$0 = \frac{\partial f^i}{\partial \dot{x}} \frac{\mathrm{d}k_i}{\mathrm{d}x_0} + \frac{\partial f^i}{\partial x} \left(S_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{ij} \frac{\mathrm{d}k_j}{\mathrm{d}x_0} \right) + \frac{\partial f^i}{\partial Z} \frac{\mathrm{d}Z_i}{\mathrm{d}x_0}, \quad i = 1, \ldots, q, \tag{2.40a}$$

$$0 = \frac{\partial g^i}{\partial x} \left(S_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{ij} \frac{\mathrm{d}k_j}{\mathrm{d}x_0} \right) + \frac{\partial g^i}{\partial Z} \frac{\mathrm{d}Z_i}{\mathrm{d}x_0}, \qquad i = 1, \ldots, q, \tag{2.40b}$$

$$S_{n+1} = S_n + T_{\mathrm{int}} \sum_{i=1}^{q} b_i \frac{\mathrm{d}k_i}{\mathrm{d}x_0}, \tag{2.40c}$$

where the following compact notation has been used for the partial derivatives $\frac{\partial f^i}{\partial \star} := \frac{\partial f}{\partial \star}(t_n + c_i T_{\mathrm{int}}, k_i, x_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i)$ and $\frac{\partial g^i}{\partial \star} := \frac{\partial g}{\partial \star}(t_n + c_i T_{\mathrm{int}}, x_n + T_{\mathrm{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i)$. Additionally, the state derivative variables $S_n = \frac{\mathrm{d}x_n}{\mathrm{d}x_0}$ are defined for which the initial value reads as $S_0 = \mathbb{1}$. These expressions can then be shown to correspond to the same IRK method, but applied to the VDAE system for the original formulation. The IND scheme therefore provides derivatives that converge to the continuous-time sensitivities with the same order as for the state trajectory [6, 44]. The equations in (2.40) are linear and they define the stage variables $\frac{\mathrm{d}k_i}{\mathrm{d}x_0}$, $\frac{\mathrm{d}Z_i}{\mathrm{d}x_0}$ for $i = 1, \ldots, q$. Note that the IFT can additionally be used on the consistency condition in (2.28d), in order to obtain the corresponding derivatives for the algebraic variables $\frac{\mathrm{d}z_n}{\mathrm{d}x_0}$ which have been omitted in (2.40). It is important to stress that these linear sensitivity equations can be used independent of the Newton-type implementation for the original nonlinear system of equations to compute the stage variables k_i and Z_i in (2.28). As will be discussed further in Section 2.4, it can however be interesting to design an efficient implementation where the Newton-type scheme and the direct IFT approach are intertwined.

We define this concept of a direct differentiation approach for implicit schemes more generally, by first introducing the following compact notation for the

integration method in (2.4) or (2.28)

$$\begin{aligned} x_{n+1} &= F(x_n, K_n, u) \\ 0 &= G(x_n, K_n, u), \end{aligned} \tag{2.41}$$

where $K_n \in \mathbb{R}^{n_K}$ refers collectively to the internal variables defined by the function $G(\cdot)$ such that the Jacobian $\frac{\partial G_n}{\partial K}(x_n, K_n, u) \in \mathbb{R}^{n_K \times n_K}$ needs to be invertible. For example, $K = (k_1, \ldots, k_q, Z_1, \ldots, Z_q)$ for a specific integration step of a q-stage RK method on a DAE system, and u forms an appropriate representation for the applied control inputs. The idea is then to obtain the first order derivatives $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0}$ and $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}u}$, based on the sensitivity results from the previous integration step, using the IFT

$$\begin{aligned} \begin{bmatrix} \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0} & \frac{\mathrm{d}x_{n+1}}{\mathrm{d}u} \end{bmatrix} &= \begin{bmatrix} \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0} & \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial F_n}{\partial u} \end{bmatrix} + \frac{\partial F_n}{\partial K} \begin{bmatrix} \frac{\mathrm{d}K_n}{\mathrm{d}x_0} & \frac{\mathrm{d}K_n}{\mathrm{d}u} \end{bmatrix} \\ \begin{bmatrix} \frac{\mathrm{d}K_n}{\mathrm{d}x_0} & \frac{\mathrm{d}K_n}{\mathrm{d}u} \end{bmatrix} &= -\frac{\partial G_n}{\partial K}^{-1} \begin{bmatrix} \frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0} & \frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial G_n}{\partial u} \end{bmatrix}, \end{aligned} \tag{2.42}$$

where the compact notation $\frac{\partial G_n}{\partial K} = \frac{\partial G}{\partial K}(x_n, K_n, u)$, $\frac{\partial G_n}{\partial x} = \frac{\partial G}{\partial x}(x_n, K_n, u)$ and $\frac{\partial G_n}{\partial u} = \frac{\partial G}{\partial u}(x_n, K_n, u)$ is used and the same holds for the derivatives of the function $F(\cdot)$. Note that the required derivative evaluations, e.g., of the form $\frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u}$ or $\frac{\partial G_n}{\partial u}$ can be obtained efficiently using AD techniques, based on either the forward or reverse mode [141]. Note that the function $F(\cdot)$ is linear when Eq. (2.41) represents an RK formula (2.4). Typically, the main computational effort in a direct approach for the sensitivity propagation of implicit integration schemes is therefore the factorization of the Jacobian $\frac{\partial G_n}{\partial K}$ and the corresponding linear system solutions [6, 259, 269].

Note that one can also use such a direct approach in combination with the reverse mode of AD in order to perform a backward sensitivity analysis. For example, one can propagate the derivatives $\lambda_n = \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_n}^\top \lambda_{n+1}$ with respect to the state value as follows

$$\begin{aligned} \lambda_n &= \frac{\partial F_n}{\partial x}^\top \lambda_{n+1} + \frac{\partial G_n}{\partial x}^\top \lambda_{n+1}^K \\ 0 &= \frac{\partial F_n}{\partial K}^\top \lambda_{n+1} + \frac{\partial G_n}{\partial K}^\top \lambda_{n+1}^K, \end{aligned} \tag{2.43}$$

where an auxiliary variable $\lambda_{n+1}^K \in \mathbb{R}^{n_K}$ has been defined. A general derivation of the above expressions can be found as part of Chapter 3. Similar to the forward set of equations in (2.42), the adjoint scheme results in a linear system which can be solved based on a factorization of the Jacobian $\frac{\partial G_n}{\partial K}$. As discussed

in [6] and unlike the case for some ERK methods, an adjoint IND scheme for an implicit integrator generally does not correspond to a simulation of the continuous-time adjoint sensitivity equations in (2.33). However, since an IND approach provides the exact derivative of the numerical simulation result, the computed adjoint derivatives converge to the continuous-time sensitivities with the same order as for the state trajectory. A special case is, for example, the implicit midpoint rule $x_{n+1} = x_n + T_{\text{int}} f_{\text{e}}(t_n + \frac{T_{\text{int}}}{2}, \frac{1}{2}(x_n + x_{n+1}))$, for which an adjoint propagation reads as

$$\lambda_n = \lambda_{n+1} + T_{\text{int}} \frac{\partial f_{\text{e}}}{\partial x}^{\top} \frac{1}{2}(\lambda_n + \lambda_{n+1}), \tag{2.44}$$

where the Jacobian denotes $\frac{\partial f_{\text{e}}}{\partial x} = \frac{\partial f_{\text{e}}}{\partial x}(t_n + \frac{T_{\text{int}}}{2}, \frac{1}{2}(x_n + x_{n+1}))$. The above expression can be derived, based on

$$\lambda_n = \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_n}^{\top} \lambda_{n+1} = \tilde{\lambda}_{n+1} + \frac{T_{\text{int}}}{2} \frac{\partial f_{\text{e}}}{\partial x}^{\top} \tilde{\lambda}_{n+1},$$

$$\text{and} \qquad \lambda_{n+1} = \tilde{\lambda}_{n+1} - \frac{T_{\text{int}}}{2} \frac{\partial f_{\text{e}}}{\partial x}^{\top} \tilde{\lambda}_{n+1}.$$

Similar to the case for the explicit RK4 method, note that the adjoint scheme (2.44) corresponds to the implicit midpoint rule applied backwards to the adjoint system $-\dot{\lambda}(t) = \frac{\partial f_{\text{e}}}{\partial x}(t, x(t))^{\top} \lambda(t)$. Such a connection between an adjoint IND scheme and the adjoint variational system of equations has also been studied numerically for BDF methods in [6].

Iterative differentiation approach

As mentioned earlier, a direct differentiation scheme relies on the assumption that the implicit equations are solved exactly. In case of a practical implementation where a limited number of Newton-type iterations are used to solve the nonlinear system, it might be advisable to instead directly differentiate the integrator implementation itself. In order to specify this alternative iterative approach to perform an IND based sensitivity propagation for implicit integration schemes, we introduce the following Newton-type method [79, 82] to solve the nonlinear system of equations $0 = G(x_n, K_n, u)$

$$K_n^{[i+1]} = K_n^{[i]} - {M^{[i]}}^{-1} G(x_n, K_n^{[i]}, u), \quad i = 0, \ldots, L-1, \tag{2.45}$$

and this for a given value of the states x_n, the control inputs u and an initial guess $K_n^{[0]}$. In this simplified representation of a Newton-type scheme, L denotes the number of iterations and the matrix $M^{[i]} \approx \frac{\partial G}{\partial K}(x_n, K_n^{[i]}, u)$ defines

a sufficiently accurate Jacobian approximation for $i = 0, \ldots, L-1$ to result in local contraction of the algorithm. As discussed earlier in Section 1.3.2, globalization strategies would generally be needed to guarantee convergence for such a Newton-type method. The integration scheme, as defined in (2.41), subsequently evaluates the next simulation result based on the computed values from the iterative scheme, i.e., $x_{n+1} = F(x_n, K_n^{[L]}, u)$.

The idea of the iterative IND scheme is then based on the direct differentiation of the Newton-type method in (2.45), including the update formula for the next result. When applying the forward mode of AD with respect to the initial state value x_0, the resulting iterative scheme reads as

$$\begin{aligned}
\frac{\mathrm{d}K_n}{\mathrm{d}x_0}^{[i+1]} &= \frac{\mathrm{d}K_n}{\mathrm{d}x_0}^{[i]} - M^{[i]^{-1}} \left(\frac{\partial G_n}{\partial K}^{[i]} \frac{\mathrm{d}K_n}{\mathrm{d}x_0}^{[i]} + \frac{\partial G_n}{\partial x}^{[i]} \frac{\mathrm{d}x_n}{\mathrm{d}x_0} \right), \; i = 0, \ldots, L-1, \\
\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0} &= \frac{\partial F_n}{\partial K}^{[L]} \frac{\mathrm{d}K_n}{\mathrm{d}x_0}^{[L]} + \frac{\partial F_n}{\partial x}^{[L]} \frac{\mathrm{d}x_n}{\mathrm{d}x_0},
\end{aligned} \tag{2.46}$$

where the compact notation $\frac{\partial G_n}{\partial K}^{[i]} = \frac{\partial G}{\partial K}(x_n, K_n^{[i]}, u)$, $\frac{\partial G_n}{\partial x}^{[i]} = \frac{\partial G}{\partial x}(x_n, K_n^{[i]}, u)$ is used and the same holds for the derivatives of the function $F(\cdot)$. Note that the directional function derivatives can be evaluated efficiently using AD techniques. Similar to the direct expressions in (2.42), one can also derive an iterative forward IND scheme to compute the derivatives with respect to the control inputs. As discussed also in [6, 44, 259], this forward IND scheme can be interpreted as an iterative procedure to solve the IFT based direct sensitivity equations in (2.42). The method therefore corresponds to an implementation of the same implicit integration formula (2.41) applied directly to the forward system of variational equations, using the same sequence of iteration matrices from the nominal simulation in (2.45). When alternatively applying the reverse mode of AD directly to the integrator, one can obtain the adjoint variant of the iterative IND scheme as described in [6, 7]. It is interesting to note that also the adjoint technique can be interpreted as a Newton-type method to solve the IFT based adjoint sensitivity equations in (2.43).

One clear advantage of the iterative scheme in (2.46) is that it merely requires the factorization of the Jacobian approximations $M^{[i]}$ for $i = 0, \ldots, L-1$, which can be reused rather easily from the original Newton-type method in (2.45). In case of the forward IND approach, both iterative schemes can be carried out simultaneously. An alternative implementation of the iterative IND technique has been proposed in [259], by using only the converged values $K_n^{[L]}$ and a corresponding Jacobian approximation from the Newton-type method. This is in contrast with the direct approach where a factorization of the exact Jacobian matrix is always needed. A quite elaborate comparison between the direct

and iterative type implementation of an IND approach including numerical experiments, can be found tailored to IRK schemes in [259, 269] and for BDF methods in [6, 7]. A direct approach for sensitivity analysis can typically be the most competitive in case the full forward sensitivity information is needed [210]. The efficient implementation of a direct IND scheme for IRK methods is discussed in the next section. This however does not exclude the benefits of an iterative scheme for the computation of relatively few forward sensitivity directions or, e.g., for an efficient adjoint sensitivity analysis. In addition, the iterative IND scheme will return in Chapter 6 and 7 in a *lifted* form as part of an inexact Newton-type optimization algorithm, where the factorization of a complete Jacobian matrix is avoided altogether.

Some interesting implementations

Note that the direct IND scheme is also referred to as the *staggered* direct method in the literature [67] because of its sequential nature by first solving the state equations iteratively in (2.45), followed by a direct solution of the IFT based sensitivity equations. On the other hand, the standard implementation of an iterative IND technique is then often called a *simultaneous corrector* method [220] where both iterative schemes are carried out simultaneously. As mentioned earlier, an alternative implementation is however possible as proposed first by [106] and which can be referred to as a staggered corrector method. The scheme also first solves the state equations, followed by a Newton-type method to iteratively solve the sensitivity equations. An overview and discussion of these three different approaches can be found in [210].

Most of the literature on discrete-time sensitivity propagation has focused on BDF methods, such as [6, 67, 106, 220]. This has led to rather popular implementations, e.g., in the software `DASSL/DASPK` [56, 220], the `DAESOL` solver [27, 99] and more recently `DAESOL-II` [7] as part of the `SolvIND` suite and `CVODES/IDAS` as part of the `SUNDIALS` package [166]. However, the first IND based schemes were proposed by [43, 48] for the semi-implicit and explicit midpoint rule, by [44, 249] and later [188, 295, 320] for Runge-Kutta methods. In addition, tailored IND based sensitivity propagation schemes have been developed for extrapolation methods in [291], for semi-implicit RK formulas in [199] and Rosenbrock-type methods in [159].

2.4 Collocation for Embedded Optimization

Even though the title suggests that only collocation methods are covered, this section further discusses some specific embedded implementation aspects which are applicable to any implicit RK formula. It should however be noted that the most popular IRK methods often belong to the family of collocation schemes, such as those based on Gauss-Legendre and Radau IIA points. In addition, a collocation method can be naturally extended with an interesting continuous output feature as presented in the next section.

2.4.1 Monitoring of Error and Convergence

Even though we focus on fixed order and step size integrators for embedded optimization throughout this thesis, it should be stressed that advanced simulation codes benefit from using adaptive integration schemes. In order to guarantee a certain numerical accuracy with a minimal computation effort by adapting the step size and order of the method, a suitable procedure for online error estimation is needed. Within the family of explicit RK methods, embedded formulas exist that provide computationally cheap error indicators by using the same function evaluations [157]. Popular examples of this consist of the Runge–Kutta–Fehlberg (RKF) and Dormand–Prince (DOPRI) methods. Such an embedded pair of methods can similarly be constructed for implicit or semi-implicit RK schemes as discussed further in [158]. Also other techniques can generally be used for error estimation such as Richardson extrapolation [81, 83], which additionally provides an extrapolated simulation result with a numerical accuracy of one order higher. Specific implementations of step size and order selection strategies can be found, for example, as part of the `LIMEX` [291] code for extrapolation methods, `DAESOL-II` [6, 7] for BDF schemes or the many solvers presented in [157, 158].

In addition to error control, advanced implementations of implicit integrators typically include a monitoring strategy on when to reuse a Jacobian approximation within the Newton-type algorithm, while preserving a good convergence rate [79, 82]. This is discussed specifically for BDF integration schemes in [7, 27]. Applying such a technique allows one to use the Jacobian information and its factorization as long as possible, in order to reduce the overall computational effort, e.g., combined with an IND based sensitivity analysis. As will be discussed further, the computation of a new Jacobian and its factorization in each integration step combined with a direct differentiation scheme can however also provide a competitive alternative implementation [199, 269]. Because of the many advantages of using these adaptive techniques for numerical

simulation, it is expected that they will soon find their way also into fast embedded implementations of dynamic optimization while respecting the real-time considerations from Section 1.5.3.

2.4.2 Newton-Type Implementations

The computational bottleneck in an implicit integration scheme is typically the solution of the nonlinear system to compute the implicitly defined variables. For this purpose, tailored Newton-type methods have been developed for different classes of integrators. Here, we mention three efficient techniques for the implementation of IRK schemes. They will be used further in this thesis when we discuss inexact Jacobian based optimization algorithms.

Reuse of Jacobian information

The first approach is based on the reuse of Jacobian information and the corresponding factorization over multiple iterations or even several integration steps, which is quite common in a practical implementation for any implicit integration scheme. Given the Newton-type method from Eq. (2.45) using the compact notation in (2.41), one would typically use the same Jacobian approximation over the L iterations such as

$$K_n^{[i+1]} = K_n^{[i]} - M^{-1} G(x_n, K_n^{[i]}, u), \quad i = 0, \dots, L-1, \tag{2.47}$$

given the invertible iteration matrix M and a corresponding factorization. Similarly, such a Jacobian approximation could be reused also over multiple integration steps when the corresponding sensitivity analysis is performed iteratively, unlike a direct differentiation technique which requires an exact Jacobian evaluation and factorization. Note that the reuse of an iteration matrix in a Newton-type method, especially over multiple integration steps, should be done cautiously in order to maintain a sufficiently fast convergence rate, e.g., by using a monitoring strategy as proposed in [7, 27, 99].

Simplified Newton iterations

The following two implementation techniques are tailored to IRK methods, exploiting the multi-stage structure of the system of nonlinear equations. For this purpose, we relate the compact notation from Eq. (2.41) and the IRK formulation from Eq. (2.28) for an implicit DAE system. The function $G(\cdot)$ which denotes the implicit equations, corresponds to the expressions in (2.28a)

and (2.28b) for an IRK method. In the case of an IRK scheme with, for example, $q = 3$ stages, the corresponding Jacobian reads as

$$\frac{\partial G}{\partial K} = \begin{bmatrix} H_1 + T_{\text{int}}\, a_{11} J_1 & T_{\text{int}}\, a_{12} J_1 & T_{\text{int}}\, a_{13} J_1 \\ T_{\text{int}}\, a_{21} J_2 & H_2 + T_{\text{int}}\, a_{22} J_2 & T_{\text{int}}\, a_{23} J_2 \\ T_{\text{int}}\, a_{31} J_3 & T_{\text{int}}\, a_{32} J_3 & H_3 + T_{\text{int}}\, a_{33} J_3 \end{bmatrix}, \tag{2.48}$$

where the matrices $H_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial z} \\ \mathbb{0} & \frac{\partial g_i}{\partial z} \end{bmatrix}$ and $J_i = \begin{bmatrix} \frac{\partial f_i}{\partial x} & \mathbb{0} \\ \frac{\partial g_i}{\partial x} & \mathbb{0} \end{bmatrix}$ have been defined. The compact notation $f_i = f(t_n + c_i T_{\text{int}}, k_i, x_n + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i)$ and $g_i = g(t_n + c_i T_{\text{int}}, x_n + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j, Z_i)$ for $i = 1, \ldots, q$ has been used for the stage function evaluations in (2.28) and the concatenated stage variables read as $K = (k_1, Z_1, \ldots, k_q, Z_q)$. A factorization of the Jacobian matrix would be needed for an exact Newton scheme to iteratively compute the stage variables in order to evaluate the integration step $x_{n+1} = x_n + T_{\text{int}} \sum_{i=1}^{q} b_i k_i$. The computational cost per iteration can be reduced considerably, based on a specific approximation of this Jacobian matrix, e.g., resulting in the *Simplified Newton* iterations as proposed originally by [35, 62].

Consider a 3-stage IRK method, for which the internal coefficient matrix $A = (a_{ij}) \in \mathbb{R}^{3\times 3}$ is assumed to be invertible. There exists a decomposition $A^{-1} = V \Gamma V^{-1}$ with a block diagonal matrix Γ. In that case, it is typical for the matrix A^{-1} to have one real eigenvalue γ and one complex conjugate eigenvalue pair $\alpha \pm i\beta$ [158]. Using the same notation, the exact Jacobian $\frac{\partial G}{\partial K}$ in (2.48) can be approximated by the following matrix

$$M = \mathbb{1}_3 \otimes H + T_{\text{int}}\, A \otimes J = \\ \begin{bmatrix} H + T_{\text{int}}\, a_{11} J & T_{\text{int}}\, a_{12} J & T_{\text{int}}\, a_{13} J \\ T_{\text{int}}\, a_{21} J & H + T_{\text{int}}\, a_{22} J & T_{\text{int}}\, a_{23} J \\ T_{\text{int}}\, a_{31} J & T_{\text{int}}\, a_{32} J & H + T_{\text{int}}\, a_{33} J \end{bmatrix}, \tag{2.49}$$

where $\otimes$ denotes the Kronecker product of matrices. We can, for example, choose the fixed Jacobians $H = H_1$ and $J = J_1$ at the first stage.

We consider the Newton-type iteration from (2.47) based on the Jacobian approximation in (2.49), which can be written as

$$(\mathbb{1}_3 \otimes H + T_{\text{int}}\, A \otimes J) \Delta K = -G, \tag{2.50}$$

where $G = G(x_n, K_n^{[i]}, u)$ denotes the function evaluation. The Newton step reads $\Delta K = K_n^{[i+1]} - K_n^{[i]}$, in the i^{th} iteration. However, premultiplying this equation on both sides by $V^{-1}(T_{\text{int}}\, A)^{-1} \otimes \mathbb{1}_{n_{\text{k}}}$ results in

$$(\tilde{\Gamma} \otimes H + \mathbb{1}_3 \otimes J) \Delta \tilde{K} = -(\tilde{\Gamma}\, V^{-1} \otimes \mathbb{1}_{n_{\text{k}}}) G, \tag{2.51}$$

where $n_\mathrm{k} = n_\mathrm{x} + n_\mathrm{z}$, $\Delta\tilde{K} = (V^{-1} \otimes \mathbb{1}_{n_\mathrm{k}})\Delta K$ and $\tilde{\Gamma} = \frac{1}{T_\mathrm{int}}\,\Gamma$. It is then important to observe that

$$\tilde{\Gamma} \otimes H + \mathbb{1}_3 \otimes J = \begin{bmatrix} \tilde{\gamma}\,H + J & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{\alpha}\,H + J & -\tilde{\beta}\,H \\ \mathbb{0} & \tilde{\beta}\,H & \tilde{\alpha}\,H + J \end{bmatrix}, \tag{2.52}$$

where the scaled eigenvalues are defined as $\tilde{\gamma} = \frac{1}{T_\mathrm{int}}\gamma$, $\tilde{\alpha} = \frac{1}{T_\mathrm{int}}\alpha$ and $\tilde{\beta} = \frac{1}{T_\mathrm{int}}\beta$. The transformed linear system (2.51) can therefore be split into two subsystems of dimension $n_\mathrm{k} = n_\mathrm{x} + n_\mathrm{z}$ and $2\,n_\mathrm{k}$. The latter $(2\,n_\mathrm{k})$-dimensional subsystem can be further transformed into a n_k-dimensional but complex subsystem as discussed in [158]. Note that this Jacobian approximation and the corresponding transformation technique can be deployed for solving both the nominal state equations and the corresponding sensitivity equations.

Single Newton iterations

A third and final Newton-type implementation for IRK schemes is often referred to as a *Single Newton* iteration, as proposed by [71, 138]. It is based on the observation that if A^{-1}, the inverse of the coefficient matrix, has only one real eigenvalue γ, then the matrix (2.52) reads as

$$\begin{bmatrix} \tilde{\gamma}\,H + J & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{\gamma}\,H + J & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \tilde{\gamma}\,H + J \end{bmatrix}. \tag{2.53}$$

This makes the linear system in (2.51) equivalent to three separate linear subsystems with the same real $n_\mathrm{k} \times n_\mathrm{k}$-matrix. Note that the computational cost of one Newton-type iteration then reduces to that of a Singly Diagonally IRK (SDIRK) method [158] even though the converged solution of the nonlinear system still corresponds to the original IRK scheme. For most high order methods, A^{-1} however does not have this property [158].

In the following, we therefore approximate the coefficient matrix by $\tilde{A}$, which is selected such that its inverse has only one real eigenvalue γ. More details on how to wisely construct such a non unique approximation can be found in [138]. The matrix $\tilde{A}$ needs to be invertible and has a decomposition of the form $\tilde{A}^{-1} = \gamma\,W(\mathbb{1}_3 - E)W^{-1}$, where E is a strictly lower triangular matrix. Using the approximate matrix $\tilde{A}$ in (2.50) and premultiplying the linear system by $W^{-1}(T_\mathrm{int}\,\tilde{A})^{-1} \otimes \mathbb{1}_{n_\mathrm{k}}$, one obtains the equivalent expression

$$\begin{aligned} (\mathbb{1}_3 \otimes (\tilde{\gamma}\,H + J))\,\Delta\hat{K} = &-(\tilde{\gamma}\,(\mathbb{1}_3 - E)\,W^{-1} \otimes \mathbb{1}_{n_\mathrm{k}})G \\ &+ (E \otimes \tilde{\gamma}\,H)\Delta\hat{K}, \end{aligned} \tag{2.54}$$

where $\tilde{\gamma} = \frac{1}{T_{\text{int}}}\gamma$ and $\Delta\hat{K} = (W^{-1} \otimes \mathbb{1}_{n_\text{k}})\Delta K$. The solution of this linear system requires the computation of just one factorization of the $n_\text{k} \times n_\text{k}$-matrix $\tilde{\gamma}\, H + J$. Since matrix E is strictly lower triangular, the linear system in (2.54) results in three separable subsystems of dimension n_k which can be solved sequentially. For example, specific Single Newton-type schemes tailored to the 3- and 4-stage Gauss method, can be found in [138, 139].

Computational complexity

The three proposed inexact Newton schemes allow for considerably reducing the computational cost of the implicit integrator. Table 2.1 shows a comparison of the computational complexity per integration step, including the presented Jacobian approximation and corresponding linear system transformation techniques. It distinguishes between the effort needed to factorize the matrix and the effort to perform the corresponding linear system solutions in each iteration. Note that the table shows the computational cost for solving the state equations and n_w forward sensitivity directions, where $n_\text{w} = n_\text{x} + n_\text{u}$ in case the full Jacobian is computed. It can be observed that the need for sensitivity analysis strongly affects the computational cost. More specifically, either the factorization or the corresponding system solutions generally become the bottleneck when using direct linear algebra, respectively if either $n_\text{k} \gg n_\text{w}$ or $n_\text{w} \gg n_\text{k}$ holds. The comparison here assumes that an LU factorization is used which, for a matrix of dimension n, requires $\sim \frac{2}{3}n^3$ flops and the back substitutions accordingly require $\sim 2n^2$ flops [137].

The table has been constructed for the Gauss collocation method, for which the coefficient matrix A has $\frac{q}{2}$ complex conjugate pairs of eigenvalues when the number of stages q is even or it has one real eigenvalue and $\frac{q-1}{2}$ complex conjugate pairs in case q is odd [158]. Note that one complex multiplication typically corresponds to 4 real multiplications, which is important for analyzing the Simplified Newton scheme. Both the Simplified and Single Newton iterations are performed with the reuse of the Jacobian as proposed in the first approach. As can be seen from Table 2.1, the Single Newton scheme is generally the cheapest to implement but it additionally approximates the coefficient matrix which typically affects the convergence as studied in more detail by [35, 62, 71]. Both the Simplified and Single Newton iterations become relatively more competitive for IRK methods with a higher number of stages [158]. The use of these inexact Jacobian based implementations within a Newton-type optimization algorithm for direct optimal control will be studied further as part of Chapter 6 and 7, including also numerical experiments.

Table 2.1: Computational cost of the Newton-type schemes per integration step for a Gauss collocation based method ($n_{\mathrm{k}} = n_{\mathrm{x}} + n_{\mathrm{z}}$ and $n_{\mathrm{w}} = n_{\mathrm{x}} + n_{\mathrm{u}}$).

	factorization (#flops)	linear system (#flops)
Exact Newton	$\frac{2}{3}(q\,n_{\mathrm{k}})^3 \times L$	$2(q\,n_{\mathrm{k}})^2(n_{\mathrm{w}}+1) \times L$
Inexact with reuse	$\frac{2}{3}(q\,n_{\mathrm{k}})^3$	$2(q\,n_{\mathrm{k}})^2(n_{\mathrm{w}}+1) \times L$
Simplified Newton	$\frac{4\,q}{3} n_{\mathrm{k}}^3$ $\frac{(4\,q-2)}{3} n_{\mathrm{k}}^3$	$(4\,q)\,n_{\mathrm{k}}^2(n_{\mathrm{w}}+1) \times L$ [q even] $(4\,q-2)\,n_{\mathrm{k}}^2(n_{\mathrm{w}}+1) \times L$ [q odd]
Single Newton	$\frac{2}{3} n_{\mathrm{k}}^3$	$(2\,q)\,n_{\mathrm{k}}^2(n_{\mathrm{w}}+1) \times L$

2.4.3 Efficient Sensitivity Propagation

Section 2.3.5 and 2.3.6 presented how to perform a tailored sensitivity analysis respectively for explicit and implicit RK formulas. We briefly discuss some of the implementation aspects, related to the two alternative sensitivity propagation techniques. One can either use an exact Jacobian evaluation and factorization for the direct computation of the necessary derivatives or implement an iterative scheme based on inexact Jacobian information. Note that sometimes a third option exists based on the use of inexact derivatives, e.g., within the Newton-type optimization algorithms discussed in Chapter 7.

Iterative differentiation

As mentioned earlier, either a staggered or simultaneous implementation of the iterative IND scheme such as in Eq. (2.46) can be made. Both variants are readily combined with the Newton-type methods for IRK schemes from the previous section. Let us summarize some of the advantages, which an iterative approach to sensitivity analysis could have also for embedded optimization and this unlike a direct IND scheme:

- It is based on the differentiation of the integrator implementation itself and therefore does not require the exact solution of the implicit equations.
- The iterative procedure can be computationally cheap in case only a few forward or backward sensitivity directions are needed.

- Any Newton-type implementation can be used and this both for the state and the sensitivity equations, possibly in a simultaneous manner. One can, e.g., reuse Jacobian information over multiple integration steps.

An efficient direct approach

Even though an iterative implementation of sensitivity analysis generally has multiple advantages, we will mostly rely on a direct computation of derivatives for optimal control throughout this thesis. The main reason is that relatively many (forward) sensitivity directions need to be computed in a standard algorithm for direct optimal control. In this particular context, an iterative solution of the sensitivity equations can be rather costly as discussed in more detail by [210, 259]. The alternative is to use a direct IND scheme which has also been referred to as a staggered direct method [6, 67]. The idea is to first solve the nonlinear system of equations corresponding to the implicit integration scheme, followed by a direct solution of the sensitivity equations. An efficient implementation of this technique is presented in Algorithm 2 for one implicit integration step of the form in (2.41), where the full Jacobian $\begin{bmatrix} \frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0} & \frac{\mathrm{d}x_{n+1}}{\mathrm{d}u} \end{bmatrix}$ of the simulation result is computed with forward differentiation as in Eq. (2.42). The algorithm is based on the reuse of the Jacobian from the sensitivity computation for the Newton-type iterations on the state equations, as proposed in [199, 269]. Note that also adjoint sensitivity propagation could be included, based on the expressions in (2.43). We summarize some of the important advantages of this direct differentiation approach:

- The direct scheme does not require an iterative procedure, but instead relies on a direct solution of the linear sensitivity equations. This forms a considerable advantage for problems with expensive function evaluations or a large number of required sensitivity directions [210].
- The Jacobian evaluation and its factorization for the sensitivity computation can be reused within the Newton-type iterations for the next integration step (see implementation in Algorithm 2).
- In order to solve the sensitivity equations exactly, a new Jacobian needs to be factorized at least once per integration step. However, this can be desirable for highly nonlinear problems, by resulting in a good Newton-type convergence for the state equations.
- When using error control for the numerical simulation (see Section 2.4.1), the sensitivities can be computed only for the accepted integration steps unlike a simultaneous iterative scheme [199].

Algorithm 2 Integration step with direct IND and Jacobian reuse (IFT-R)

Input: (x_n, u), $\frac{\mathrm{d}x_n}{\mathrm{d}(x_0,u)}$, initial guess $K_n^{[0]}$ and factorized M.

`Newton-type scheme` (2.47)

1: **for** $i = 0 \to L-1$ **do**

2: $\quad K_n^{[i+1]} \leftarrow K_n^{[i]} - M^{-1} G(x_n, K_n^{[i]}, u)$.

3: **end for**

4: $x_{n+1} \leftarrow F(x_n, K_n^{[L]}, u)$.

`Evaluate and factorize Jacobian`

5: $M \leftarrow \frac{\partial G}{\partial K}(x_n, K_n^{[L]}, u)$.

`Derivatives collocation variables`

6: $\frac{\mathrm{d}K_n}{\mathrm{d}x_0} \leftarrow -M^{-1}\left(\frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0}\right)$.
$\frac{\mathrm{d}K_n}{\mathrm{d}u} \leftarrow -M^{-1}\left(\frac{\partial G_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial G_n}{\partial u}\right)$.

`Forward sensitivities result`

7: $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}x_0} \leftarrow \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}x_0} + \frac{\partial F_n}{\partial K}\frac{\mathrm{d}K_n}{\mathrm{d}x_0}$.
$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}u} \leftarrow \frac{\partial F_n}{\partial x}\frac{\mathrm{d}x_n}{\mathrm{d}u} + \frac{\partial F_n}{\partial K}\frac{\mathrm{d}K_n}{\mathrm{d}u} + \frac{\partial F_n}{\partial u}$.

8: Initialize $K_{n+1}^{[0]}$ for next integration step [259].

Output: x_{n+1}, $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}(x_0,u)}$, next guess $K_{n+1}^{[0]}$ and factorized M.

The forward sensitivity propagation in Algorithm 2 requires the computation of the quantities $\frac{\partial F_n}{\partial K}\frac{\mathrm{d}K_n}{\mathrm{d}x_0}$ and $\frac{\partial F_n}{\partial K}\frac{\mathrm{d}K_n}{\mathrm{d}u}$, based on the factorization of the Jacobian matrix M. As discussed in [259], it can be computationally more efficient to first evaluate the expression $\frac{\partial F_n}{\partial K}M^{-1}$ in case of many sensitivity directions, i.e., when $(n_\mathrm{x} + n_\mathrm{u}) \gg n_\mathrm{x}$. This alternative approach bears resemblance to an adjoint sensitivity analysis. It is however not attractive anymore when many extra outputs need to be computed, as discussed in the next section. Note that Algorithm 2 has been referred to as the IFT based scheme with reuse of the Jacobian, abbreviated IFT-R in [259, 269]. In these references, a detailed comparison with other IND implementations can be found for collocation methods, based on numerical experiments.

Remark 2.4 *Most real-time optimal control implementations are based on tailored structure exploiting optimization and simulation algorithms, such as the techniques presented in Chapter 4, 5 and 6. They require efficient direct linear algebra routines for small to medium-scale problems. Most general-purpose implementations of the Basic Linear Algebra Subprograms (BLAS) standard are however not optimized for these problem sizes, such that more efficient custom implementations can be made for embedded optimization. For this, we refer the reader to [121, 123] and references therein.*

2.5 Continuous Output for Optimal Control

An important property of numerical simulation methods is whether they allow for a continuous representation of the solution to the system of differential equations. For example, the family of collocation methods naturally provides this feature. But also other schemes can be extended with such a *dense* or *continuous output* formula. This concept can be practically relevant to obtain graphical output, for event localization or the treatment of discontinuities in differential equations [157]. An interesting example of the latter is the detection of implicit switches between different sub-models as discussed in [55, 188]. Continuous output could be used for the implementation of adjoint sensitivity analysis, in order to efficiently interpolate the forward solution while performing the adjoint model simulation [15].

The feature of continuous output can also be computationally attractive to be used within algorithms for direct optimal control, in order to efficiently define certain objective or constraint functions. Let us mention some examples of optimal control related applications:

- For the accurate numerical evaluation of continuous objectives [188].
- To impose path constraints on a grid, that can be (much) finer than the discretization grid. This can, for example, be used to efficiently implement closed-loop costing for NMPC as discussed in Section 2.5.3.
- In order to evaluate the simulation results, corresponding to high frequency measurements for dynamic estimation problems (see Section 2.5.4).
- It can also be used for the parameterization of coupling variables in a distributed optimal control framework as presented in Chapter 5.

2.5.1 Continuous Extension of Integration Formulas

Note that the alternative to the continuous extension of a numerical simulation method, is to guarantee the integration step size to be sufficiently small corresponding to the required resolution for the output evaluation grid. This is however typically undesirable because the resulting simulation procedure would become computationally inefficient. One of the first dense output schemes was constructed for the Runge–Kutta–Fehlberg (RKF) formula of order 4-5 by [171]. Eventually, the large family of RK methods can be divided in a continuous and a discrete group [102]. Schemes in the first group naturally provide a continuous representation of the numerical solution, such as the class

of collocation methods which is discussed in the next section. But also BDF methods can naturally and efficiently provide a continuous output by using interpolation polynomials as discussed in [47, 55]. An overview on how to generally construct continuous extensions for the other RK methods can be found in [101, 102] and references therein. Multiple techniques can be used that differ in their applicability, the continuity of the interpolant across integration steps and the resulting computational cost, which is proportional to the required amount of extra function evaluations.

2.5.2 Family of Collocation Methods

Each q-stage collocation method defines a polynomial which is a continuous representation for the numerical simulation of order q as shown in [157]. Note that the end point order of the scheme can still be constructed to be higher by selecting the collocation nodes, e.g., $P = 2\,q$ for the Gauss methods [158]. This property for the Gaussian quadrature formula is sometimes referred to as superconvergence. Based on the q^{th} order collocation polynomial $p(t)$ in Eq. (2.17), the corresponding continuous output evaluation results in a local interpolation error $\mathcal{O}(T_{\text{int}}^{q+1})$. As argued in more detail by [70], this relation however also holds for the global error resulting in a higher order of $q+1$ in case the end point order $P > q$. Therefore, the continuous approximation of the state trajectory is of order $\tilde{P} = \min(P, q+1)$ for a q-stage collocation method. More specifically, Table 2.2 presents the end point P and continuous output order $\tilde{P}$ for the first three Gauss (GL2, GL4 and GL6) and Radau IIA (RIIA1, RIIA3 and RIIA5) methods with $q = 1$, 2 and 3 stages [158]. Even though the class of collocation methods provides a natural continuous extension without extra function evaluations, a clear and undesirable discrepancy can be observed in the order of accuracy for higher order methods.

		RIIA1	GL2	RIIA3	GL4	RIIA5	GL6
number of stages	q	1	1	2	2	3	3
end point order	P	1	2	3	4	5	6
continuous output	$\tilde{P}$	1	2	3	3	4	4

Table 2.2: Order of accuracy for the end point and of the continuous output for the 1-, 2- and 3-stage Gauss-Legendre and Radau IIA methods.

Continuous output evaluation

Collocation methods form a specific family of IRK schemes as in (2.4), for which their continuous extension arises naturally. Using the state derivative variables $k_1, \ldots, k_q$, a polynomial interpolation $p(t)$ is defined for the state trajectory as in (2.17) over the interval $t \in [t_n, t_{n+1}]$

$$x(t_n + c\,T_{\text{int}}) \approx x_n + T_{\text{int}} \sum_{i=1}^{q} k_i \int_0^c \ell_i(\tau)\,\mathrm{d}\tau, \tag{2.55}$$

at a specific time point $t_n + c\,T_{\text{int}}$, where $0 \leq c \leq 1$ and $\ell_i(t) = \prod_{j \neq i} \frac{t - c_j}{c_i - c_j}$ are the Lagrange interpolating polynomials as in (2.19). Note that for $c = 1$, this corresponds to the evaluation of $x_{n+1} = x_n + T_{\text{int}} \sum_{i=1}^{q} b_i k_i$ based on the coefficients in (2.18). As mentioned earlier, this continuous output evaluation can be shown to be of order $\tilde{P} = \min(P, q+1)$.

It is possible to also construct such a polynomial representation for the algebraic states $z(t)$ and differential state derivatives $\dot{x}(t)$, in the case of a collocation method (2.28) applied to a DAE system. As discussed also in [259, 261], in addition to the computed stage variables $K = (k_1, \ldots, k_q, Z_1, \ldots, Z_q)$, this would also require consistent values of the variables at time t_n in order to obtain the same order of accuracy $\tilde{P}$ as for the differential state trajectory. In the case of a discontinuous jump in the value of a parameter or control input, the differential states vary continuously while their derivatives and the algebraic states can also exhibit such a jump as a result. In practice, the assumption of having consistent values for these variables at the beginning of each integration step is therefore not always evident. An interpolating polynomial of one order less could then be used for these variables

$$\begin{aligned} z(t_n + c\,T_{\text{int}}) &\approx \sum_{i=1}^{q} \ell_i(c) Z_i, \\ \dot{x}(t_n + c\,T_{\text{int}}) &\approx \sum_{i=1}^{q} \ell_i(c) k_i. \end{aligned} \tag{2.56}$$

Therefore, the differential as well as the algebraic variables and state derivatives can efficiently be evaluated on any set of grid points which can be chosen independently from the discretization grid. In the context of an optimal control problem formulation, there would typically be an output function of the form $y(t) = \psi(t, \dot{x}(t), x(t), z(t))$ which needs to be evaluated on a fine grid. This can be performed by using the polynomial interpolations in Eqs. (2.55) and (2.56). Even though continuous output can be used as a feature to efficiently implement

adjoint sensitivity analysis, note that it is conversely not a good idea to use adjoint sensitivity propagation to compute first order derivative information for a relatively high dimensional output function. Instead, one could directly evaluate the output sensitivities using the forward derivatives of the collocation variables, e.g., from Algorithm 2.

2.5.3 Closed-Loop Costing based on Continuous Output

The issue of closed-loop stability for an NMPC scheme has briefly been discussed in Section 1.5 of the previous chapter. Instead of determining a suitable terminal region or sufficiently prolonging the control horizon [17], we look into the technique of *infinite horizon closed-loop costing* [219, 231] as an example for the use of continuous output within algorithms for direct optimal control. The scheme is based on a local control law and uses a prediction horizon in which the state and input constraints are still imposed. Stability of the closed-loop system can then be proven in a rigorous way [231]. To formulate a practical scheme, the prediction horizon can be truncated without losing this stability guarantee as is the topic of discussion in [219]. Furthermore, the path constraints will only be imposed at specific time points. The main advantage of introducing the prediction horizon is that it allows us to enlarge the region of attraction around a reference point without increasing the control horizon, i.e., with the same number of decision variables [93].

An efficient implementation of the approximate infinite horizon closed-loop costing approach has been presented in [271], in order to implement a stable NMPC scheme for the example of an inverted pendulum system on top of a cart. In this tutorial case study, an LQR controller was designed as the local control law around the unstable steady state reference point. In addition, the collocation method from Algorithm 2 has been used, including the continuous output feature with forward sensitivity propagation as detailed in the previous section. An efficient implementation of this scheme can be found in the open-source `ACADO` code generation tool, as discussed in [271] but this software is also presented further in Chapter 8. The goal of the integrator is to take relatively large integration steps, but to use continuous output in order to efficiently evaluate the additional path constraints over the prediction horizon as well as the corresponding terminal cost. The numerical results for an RTI implementation can be found in [271], based on a Gauss method of order 4 ($q = 2$), integration step size $T_{\text{int}} = 0.05$ s, sampling time $T_{\text{s}} = 0.05$ s and using `FORCES` to solve the structured QP subproblem at each time step. Table 2.3 shows the average computation times, both for a formulation with and without a prediction horizon. It can be observed that the closed-loop costing scheme

Table 2.3: Average computation time for approximate infinite horizon NMPC.[1]

	$(N_c = 20,\ N_p = 0)$	$(N_c = 10,\ N_p = 20)$
Simulation and sensitivities:		
• control horizon	60 µs	30 µs
• prediction horizon	-	14 µs
QP solution (`FORCES`)	155 µs	114 µs
Total RTI step	215 µs	158 µs

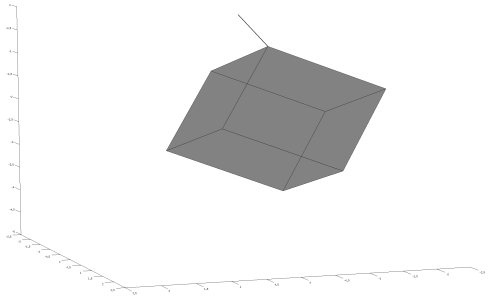


Figure 2.6: The cube toy example setup, used in the multi-rate estimation problem to illustrate a continuous output based MHE implementation.

based on continuous output can provide a rather efficient way to improve the stability properties of the NMPC implementation.

2.5.4 MHE with Multi-Rate Measurements

As a final motivating example for the use of continuous output within direct optimal control, we look at an MHE scheme with multi-rate measurements as presented in [261]. Figure 2.6 depicts the setup that is considered as a toy example, which consists of a cube hanging to a rod by one of its corners. The corresponding index-1 DAE model can be obtained from Lagrange mechanics as described in more detail in [261]. The goal of the problem is to accurately estimate the position and orientation of the moving cube, using two different types of measurements [128]. Absolute location information is available at a

[1] These numerical simulations are performed using the `ACADO` code generation tool on a computer equipped with Intel i7-3720QM processor, running a 64-bit version of Ubuntu 12.04.

	ACADO solver	MATLAB's ode15s
Simulation and sensitivities	0.02 s	10.65 s
Total time per CGN iteration	0.04 s	10.69 s

Table 2.4: Average computation time of one CGN iteration, using respectively an auto generated ACADO solver and MATLAB's general-purpose DAE solver ode15s.[2]

low frequency of 10 Hz using a Global Positioning System (GPS). In addition, measurements of the angular velocity and linear acceleration are provided by an Inertial Measurement Unit (IMU) at a relatively high frequency of, e.g., 800 Hz.

The resulting OCP formulation consists of an equality constrained nonlinear optimization problem with a least squares objective. Let us consider a number of estimation intervals $N = 10$ and the corresponding horizon length of $T = 1$ s. In order to implement the MHE scheme, this OCP can be solved at each time point using multiple shooting in combination with the Constrained Gauss-Newton (CGN) method from [44]. Similar to the previous numerical example, we can use the ACADO generated collocation integrators with continuous output and forward sensitivity propagation. This allows one to use an integration step size that is significantly larger than the time between two IMU measurements without any loss of information, by evaluating the corresponding simulation results on the fine grid using continuous output. As an alternative approach, the same CGN algorithm can be implemented using a standard DAE solver from MATLAB such as ode15s, which also supports the evaluation of additional outputs. Table 2.4 shows the computation times for both implementations, where numerical differentiation is used to generate the sensitivities for the ode15s solver. The table illustrates the importance of using an integration scheme with tailored sensitivity propagation for direct optimal control. It is however important to note that both implementations are quite different and difficult to compare directly as discussed in [261].

2.6 Conclusions and Outlook

This chapter provided an overview on numerical integration schemes and tailored first order sensitivity analysis for the efficient implementation of embedded optimization algorithms for direct optimal control. The efficient implementation

[2]The numerical experiments were run on an ordinary computer (Intel P8600 3MB cache, 2.40 GHz, 64-bit Ubuntu 12.04 LTS and MATLAB R2011a 64 bit).

of Implicit Runge-Kutta (IRK) methods for embedded optimization is discussed in more detail. For this purpose, one can use tailored Newton-type methods in addition to the reuse of Jacobian factorizations for both the numerical simulation and the sensitivity propagation. A promising feature of collocation methods, which form a subclass of IRK schemes, is that they can provide continuous output evaluations at an additional computational cost which is relatively small. The use of continuous output for direct optimal control has been illustrated, based on the concept of closed-loop costing for NMPC and the efficient implementation of MHE with multi-rate measurements.

The next chapters will build further on these proposed algorithmic techniques, e.g., in order to include a tailored exploitation of particular dynamic system structures in Chapter 4 and 5, as well as the efficient propagation of second order sensitivities for Newton-type optimization in Chapter 3.

Chapter 3

Symmetric Hessian Propagation Technique

This chapter proposes an efficient scheme for both discrete- and continuous-time second order sensitivity propagation within direct methods for optimal control. Unlike the classical forward-over-adjoint (FOA) techniques [141, 240] to compute second order derivative information, we propose a novel Hessian propagation technique to maintain and exploit the symmetric property of Hessian computations in Newton-type optimization. The scheme is presented in combination with any explicit or implicit integration method. In addition, we present an extension of these results to continuous-time sensitivity propagation for an implicit system of Differential-Algebraic Equations (DAE) of index 1. This discussion in a continuous-time framework allows for a generic sensitivity analysis, before applying a numerical discretization scheme.

Based on the symmetric sensitivity equations, a three-sweep Hessian propagation (TSP) scheme is proposed. This novel technique is studied here both in discrete- and continuous-time, and shown to considerably reduce both the computational burden and the memory requirements over classical approaches. In the case of adjoint or second order sensitivity analysis, one important issue is the memory footprint for the resulting propagation scheme consisting of consecutive forward and backward sweeps. With the use of this TSP technique based on the symmetric sensitivity equations, one can reduce the storage requirements. An implementation of these symmetric Hessian propagation techniques in the open-source `ACADO Toolkit` software package is illustrated numerically on the case study of a nonlinear biochemical reactor.

Note that this chapter is largely based on the article in [266], following the initial results in [267] and [268].

Notation and preliminaries This thesis denotes first order total and partial derivatives, respectively using the compact notation $\mathrm{D}_a F(a,b) = \frac{\mathrm{d}F(a,b)}{\mathrm{d}a}$ and $\partial_a F(a,b) = \frac{\partial F(a,b)}{\partial a}$. In addition, let us write the second order directional derivatives:

$$\langle c, \mathrm{D}^2_{a,b} F(a,b) \rangle = \sum_{k=1}^{n} c_k \frac{\mathrm{d}^2 F_k(a,b)}{\mathrm{d}a \, \mathrm{d}b}, \tag{3.1}$$

where $c \in \mathbb{R}^n$ is a constant vector and $F : \mathbb{R}^l \times \mathbb{R}^m \to \mathbb{R}^n$ is a twice differentiable function. Notice that the map $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^{n \times l \times m} \to \mathbb{R}^{l \times m}$ does not denote a standard scalar product, since the first argument is a vector while the second argument is a tensor. We occasionally use the shorthand notation $\langle c, \mathrm{D}^2_{a,b} F \rangle$, if it is clear from the context that F depends on a and b. We write $\langle c, \mathrm{D}^2_a F(a) \rangle$ rather than $\langle c, \mathrm{D}^2_{a,a} F(a) \rangle$, in case F has only one argument. If the second order derivatives of F are continuous, the matrix $\langle c, \mathrm{D}^2_a F(a) \rangle$ is symmetric. When using partial instead of total derivatives, a similar compact notation for the directional second order derivatives is adopted:

$$\langle c, \partial^2_{a,b} F(a,b) \rangle = \sum_{k=1}^{n} c_k \frac{\partial^2 F_k(a,b)}{\partial a \, \partial b}. \tag{3.2}$$

When it is clear whether they are the result of partial or total differentiation, first and second order derivatives can sometimes be denoted more compactly as, e.g., $F_a = \frac{\partial F(a,b)}{\partial a}$, respectively $F_{a\,a} = \frac{\partial^2 F(a,b)}{\partial a^2}$ or $F_{a\,b} = \frac{\partial^2 F(a,b)}{\partial a \, \partial b}$.

Outline The chapter is organized as follows. Section 3.1 briefly introduces a simplified problem formulation in order to illustrate the need for efficient sensitivity analysis within direct optimal control. Section 3.2 discusses discrete-time propagation techniques for a generic implicit integration method. We first present the classical first and second order techniques, then we propose and motivate our alternative symmetric propagation scheme. Section 3.3 then presents the continuous-time extension of these novel sensitivity equations, considering an implicit DAE system. The three-sweep Hessian propagation technique is introduced and discussed in Section 3.4, including implementation aspects. Section 3.5 finally presents numerical results on an illustrative case study, using the open-source `ACADO Toolkit` software.

3.1 Problem Statement

Let us briefly introduce the problem formulation in which we are interested, including the DAE system and the need for first and second order sensitivity analysis for Newton-type optimization within direct optimal control.

3.1.1 Differential-Algebraic Equations

We consider the following semi-explicit DAE system, in which we omitted the dependency on control inputs in Eq. (1.5) for simplicity of notation:

$$\begin{aligned} \dot{x}(t) &= f(x(t), z(t)), \quad x(0) = x_0(p), \\ 0 &= g(x(t), z(t)), \end{aligned} \tag{3.3}$$

where $x(t) \in \mathbb{R}^{n_x}$ denotes the differential states, $z(t) \in \mathbb{R}^{n_z}$ the algebraic variables and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. The parameters $p \in \mathbb{R}^{n_p}$ are additional variables that define the initial value $x_0 : \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$. The DAE system is of index 1 (see Definition 1.4) if the Jacobian $\partial_z g(\cdot)$ is non-singular. When there are no algebraic variables, the set of equations instead denotes an explicit ODE system:

$$\dot{x}(t) = f_{\text{ODE}}(x(t)), \quad x(0) = x_0(p). \tag{3.4}$$

We introduce the following two important assumptions.

Assumption 3.1 *The functions $f(x(t), z(t))$, $g(x(t), z(t))$ and $x_0(p)$ are twice continuously differentiable in all arguments (see also Assumption 1.10).*

Assumption 3.2 *The DAE system* (3.3) *has differential index* 0 *or index* 1*, which means that either $n_z = 0$, or the Jacobian matrix $\partial_z g(\cdot)$ must be invertible.*

Finally, we refer to the following definition for consistent initial conditions such that the initial value problem in Eq. (3.3) has a unique solution $x(t, p)$ and $z(t, p)$ $\forall t \in [0, T], p$ given the previous two assumptions [67, 158, 279, 280].

Definition 3.3 (Consistent initial conditions) *The values $x(0, p)$, $z(0, p)$ are called consistent when the following conditions hold:*

$$\begin{aligned} x(0, p) &= x_0(p) \\ 0 &= g(x(0, p), z(0, p)). \end{aligned} \tag{3.5}$$

This is a well defined nonlinear system in the variables $x(0, p)$, $z(0, p)$ given the parameter values p and an index 1 *DAE.*

Note that the sensitivity propagation techniques presented in this chapter can be readily extended to the fully implicit DAE formulation from Definition 1.3. To keep the notation relatively simple, let us however focus on the semi-explicit formulation from Eq. (3.3) instead.

3.1.2 Direct Optimal Control

Based on the initial value problem from Eq. (3.3), we consider the following continuous-time OCP formulation

$$\min_{x(\cdot),z(\cdot),p} \quad m\left(x(T)\right) \tag{3.6a}$$

$$\text{s.t.} \qquad 0 = x(0) - x_0(p), \tag{3.6b}$$

$$\dot{x}(t) = f(x(t), z(t)), \ \forall t \in [0,T], \tag{3.6c}$$

$$0 = g(x(t), z(t)), \ \forall t \in [0,T], \tag{3.6d}$$

where the objective in Eq. (3.6a) consists of a terminal cost defined by the twice continuously differentiable function $m(\cdot)$, depending only on the differential states for notational convenience. For the ease of exposition, note that this is a simplified form of the OCP in Eq. (1.6), which could comprise additional time-varying control inputs, inequality constraints on states and controls, or more general objective functionals.

The unique solution of the initial value problem in Eq. (3.3) can be referred to as $x(t,p)$ and $z(t,p)$ $\forall t \in [0,T]$ and for a specific parameter value p. As discussed earlier in Section 1.2.2, a single shooting type discretization of the OCP (3.6) then results in the following unconstrained NLP:

$$\min_{p} \quad \mathcal{M}(p), \tag{3.7}$$

where $\mathcal{M}(p) = m\left(x(T,p)\right)$. A minimizer for this finite dimensional problem exists if and only if the continuous-time OCP (3.6) has an optimal, bounded solution. To preserve a more compact notation, single shooting will be used throughout this chapter even though all presented techniques can also readily be employed within the direct multiple shooting method [48]. Note that direct transcription methods [34] do not require a propagation of sensitivities as in shooting based approaches, but they can still benefit from the proposed symmetric evaluation of the Hessian contributions. In practice, the function $x(T,p)$ is obtained approximately by the use of a numerical simulation scheme as discussed in the previous chapter.

3.1.3 Newton-Type Optimization

Exact Newton methods of the form

$$p^{[k+1]} = p^{[k]} - \mathrm{D}_p^2 \mathcal{M}(p^{[k]})^{-1} \, \mathrm{D}_p \mathcal{M}(p^{[k]})^\top, \quad \text{for } k = 0, 1, \ldots \tag{3.8}$$

converge locally quadratically to stationary points of problem (3.7) under mild conditions; see Section 1.3.2. In general, one needs a well designed globalization strategy to guarantee convergence, but this topic is outside of the scope here. The first and second order derivatives in Eq. (3.8) are given by:

$$\begin{aligned} \mathrm{D}_p \mathcal{M}(p)^\top &= \mathrm{D}_p x(T,p)^\top \bar{\lambda}(p) \\ \mathrm{D}_p^2 \mathcal{M}(p) &= \langle \bar{\lambda}(p), \mathrm{D}_p^2 x(T,p) \rangle + \mathrm{D}_p x(T,p)^\top \partial_x^2 m\left(x(T,p)\right) \, \mathrm{D}_p x(T,p), \end{aligned} \tag{3.9}$$

where the notation $\bar{\lambda}(p)^\top = \partial_x m\left(x(T,p)\right)$ is used. Here, $\bar{\lambda}(p)$ is called the "backward seed" for the sensitivity propagation techniques. The evaluation of partial derivatives for the function $m(\cdot)$ is assumed to be relatively cheap using AD techniques as discussed in Section 2.3.1 of the previous chapter. The main computational effort is therefore typically the numerical simulation to evaluate $x(T,p)$ and the propagation of its first and second order sensitivities. Note that these directional second order derivatives $\langle \bar{\lambda}(p), \mathrm{D}_p^2 x(T,p) \rangle = \sum_{k=1}^{n_x} \bar{\lambda}_k(p) \mathrm{D}_p^2 x_k(T,\, p)$ are defined as in (3.1).

3.2 Discrete-Time Sensitivity Propagation

For a discussion on discrete-time propagation techniques for the sensitivities in (3.9), let us denote an integration scheme to simulate the index-1 DAE system in Eq. (3.3) by the following semi-explicit set of equations:

$$\begin{aligned} x_{n+1} &= F(x_n, z_n) \\ 0 &= G(x_n, z_n), \end{aligned} \tag{3.10}$$

for $n = 0, \ldots, N_\mathrm{s} - 1$ where $x_0 = x_0(p)$, the Jacobian matrix $\partial_z G(\cdot)$ is invertible and the functions $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_x}$, $G : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ are twice continuously differentiable when Assumption 3.1 holds. In what follows, the dependency of the variables x_n, z_n on the parameter value p for $n = 0, \ldots, N_\mathrm{s} - 1$ will be omitted to arrive at a compact notation.

Note that the formulation in Eq. (3.10) can be considered a simplified form of Eq. (2.41) which includes both the explicit and implicit integration methods

that have been presented in the previous chapter, where the function $G(\cdot)$ remains empty in case of an explicit scheme. The additional variables $z_n \in \mathbb{R}^{n_z}$ for $n = 0, \dots, N_\mathrm{s} - 1$ denote all internal variables that are necessary to formulate the integration scheme. In the case of a collocation method, for example, these variables denote the values of the differential state derivatives and algebraic variables at the collocation nodes such as in Eq. (2.28). The number of integration steps N_s is chosen to be fixed for notational convenience. This section is concerned with a discrete-time sensitivity analysis, following the discretize-then-differentiate principle as introduced in Section 2.3.2. For this purpose, a direct differentiation approach will be adopted even though the same techniques can be readily used to derive an iterative propagation scheme as illustrated earlier in Section 2.3.6 for the IRK methods.

3.2.1 First Order Sensitivity Analysis

In what follows, we will refer to the function evaluations in Eq. (3.10) using the compact notation $F^n = F(x_n, z_n)$ and $G^n = G(x_n, z_n)$. Similarly, the first order derivatives read:

$$F_x^n = \partial_x F^n, \quad F_z^n = \partial_z F^n \quad \text{and} \quad G_x^n = \partial_x G^n, \quad G_z^n = \partial_z G^n, \tag{3.11}$$

of which the Jacobian matrix G_z^n is invertible. Before we discuss second order derivatives and present our novel symmetric algorithm, let us recall the first order sensitivity analysis for the integration scheme in (3.10).

Forward propagation We define the sensitivities $S_n^x := \mathrm{D}_p x_n \in \mathbb{R}^{n_x \times n_p}$ and $S_n^z := \mathrm{D}_p z_n \in \mathbb{R}^{n_z \times n_p}$, which can be propagated forward using the following semi-explicit system of equations:

$$\begin{aligned} S_{n+1}^x &= F_x^n \, S_n^x + F_z^n \, S_n^z \\ 0 &= G_x^n \, S_n^x + G_z^n \, S_n^z, \end{aligned} \tag{3.12}$$

for $n = 0, \dots, N_\mathrm{s} - 1$ where the initial value $S_0^x = \mathrm{D}_p x_0$ is known. The sensitivity equations in (3.12) can be obtained directly by differentiating Eq. (3.10) with respect to the parameter p. Note that the matrix G_z^n is invertible such that $S_n^z = -{G_z^n}^{-1} G_x^n \, S_n^x$ could be computed explicitly. The end value $S_{N_\mathrm{s}}^x$ can be used to obtain the result $\mathrm{D}_p \mathcal{M}(p)^\top = \mathrm{D}_p (x_{N_\mathrm{s}})^\top \, \bar{\lambda} = {S_{N_\mathrm{s}}^x}^\top \bar{\lambda}$ in the Newton-type optimization scheme from Section 3.1.3.

Adjoint propagation The gradient result $\mathrm{D}_p \mathcal{M}(p)^\top$ can alternatively be computed directly by use of an adjoint propagation scheme where $\bar{\lambda}$ denotes

the backward seed. For this purpose, let us define the adjoint variables $\lambda_n^x := \mathrm{D}_{x_n}(x_{N_\mathrm{s}})^\top \bar{\lambda} \in \mathbb{R}^{n_x}$ and $\lambda_n^z \in \mathbb{R}^{n_z}$ that can be propagated backward using the following semi-explicit system of equations:

$$\begin{aligned} \lambda_n^x &= {F_x^n}^\top \lambda_{n+1}^x + {G_x^n}^\top \lambda_{n+1}^z \\ 0 &= {F_z^n}^\top \lambda_{n+1}^x + {G_z^n}^\top \lambda_{n+1}^z, \end{aligned} \tag{3.13}$$

for $n = N_\mathrm{s} - 1, \ldots, 0$ where the initial value $\lambda_{N_\mathrm{s}}^x = \bar{\lambda}$ is given by the seed. The backward propagation scheme results in the sensitivity $\lambda_0^x = \mathrm{D}_{x_0}(x_{N_\mathrm{s}})^\top \bar{\lambda}$ such that $\mathrm{D}_p \mathcal{M}(p)^\top = \mathrm{D}_p(x_0)^\top \lambda_0^x$. These adjoint sensitivity equations (3.13) can be obtained directly by differentiating (3.10) with respect to x_n and multiplying the first equation with $\lambda_{n+1}^{x^\top}$:

$$\mathrm{D}_{x_n}(x_{n+1})^\top \lambda_{n+1}^x = {F_x^n}^\top \lambda_{n+1}^x + \mathrm{D}_{x_n}(z_n)^\top {F_z^n}^\top \lambda_{n+1}^x$$

$$0 = {G_x^n}^\top + \mathrm{D}_{x_n}(z_n)^\top {G_z^n}^\top,$$

which defines $\mathrm{D}_{x_n}(z_n)^\top = -{G_x^n}^\top {G_z^n}^{-\top}$. By introducing $\lambda_{n+1}^z = -{G_z^n}^{-\top} {F_z^n}^\top \lambda_{n+1}^x$, one obtains the expressions in Eq. (3.13).

3.2.2 Second Order Sensitivity Propagation

Next, we are interested in computing second order directional derivatives of the form $\langle \bar{\lambda}, \mathrm{D}_p^2 x_{N_\mathrm{s}} \rangle = \langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$ as required in Eq. (3.9) and following our notation in Eq. (3.1). Such directional second order derivatives can be computed by combining forward and backward techniques for first order sensitivity analysis. Combining the two techniques for first order derivatives results in four possible propagation schemes [141, 240]. However, in the *forward-over-forward* approach, computational effort would be spent in computing sensitivity directions that are not necessarily needed to form the Hessian result. Similarly, it is not efficient to perform more than one reverse sweep as discussed in [141]. In the following, among the two remaining approaches, preference will be given to the more standard *forward-over-adjoint* (FOA) approach.

We introduce the following compact notation for the second order derivatives of the functions $F^n = F(x_n, z_n)$ and $G^n = G(x_n, z_n)$:

$$F_{a\,b}^n = \partial_{a,b}^2 F^n \quad \text{and} \quad G_{a\,b}^n = \partial_{a,b}^2 G^n. \tag{3.14}$$

Forward-over-adjoint (FOA) propagation Let us apply forward differentiation directly to the adjoint propagation scheme in Eq. (3.13), where we also regard the

dependency of the variables λ_n on the parameter p. This results in the additional variables $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$, for which the corresponding FOA type equations read as:

$$\begin{aligned} H_n^x &= {F_x^n}^\top H_{n+1}^x + {G_x^n}^\top H_{n+1}^z \\ &\quad + \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix} \\ 0 &= {F_z^n}^\top H_{n+1}^x + {G_z^n}^\top H_{n+1}^z \\ &\quad + \begin{bmatrix} \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \end{aligned} \tag{3.15}$$

for $n = N_\mathrm{s} - 1, \ldots, 0$ where the initial value is given by $H_{N_\mathrm{s}}^x = \mathbb{0}$. This backward propagation results in the sensitivity $H_0^x = \langle \bar{\lambda}, \mathrm{D}^2_{x_0,p} x_{N_\mathrm{s}} \rangle$ such that the Hessian matrix $\langle \bar{\lambda}, \mathrm{D}_p^2 x_{N_\mathrm{s}} \rangle = {H_0^x}^\top \mathrm{D}_p x_0 + \langle \lambda_0^x, \mathrm{D}_p^2 x_0(p) \rangle$ can be evaluated. It is important to note that the FOA variables $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$ are not symmetric or even square, and the same holds for the equations in the FOA type propagation scheme (3.15). The desired Hessian result $\langle \bar{\lambda}, \mathrm{D}_p^2 x_{N_\mathrm{s}} \rangle$ is however symmetric by definition.

3.2.3 Symmetric Second Order Sensitivity Propagation

We are here interested in an approach that can efficiently propagate a symmetric Hessian variable $H_n^\mathrm{S} \in \mathbb{R}^{n_p \times n_p}$ directly. In what follows, we show that such a symmetric propagation scheme provides multiple benefits over the classical FOA approach, while both provide the same second order sensitivities. The following theorem summarizes this result.

Theorem 3.4 (Symmetric Hessian propagation) *Let* (x_{n+1}, z_n), (S_{n+1}^x, S_n^z) *and* $(\lambda_n^x, \lambda_{n+1}^z)$ *be defined for* $n = 0, \ldots, N_\mathrm{s} - 1$ *respectively by Eqs.* (3.10), (3.12) *and* (3.13) *and the corresponding initial and terminal values. The following propagation scheme then generates a symmetric variable* H_n^S*:*

$$\begin{aligned} H_{n+1}^\mathrm{S} &= H_n^\mathrm{S} \\ &\quad + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \end{aligned} \tag{3.16}$$

for $n = 0, \ldots, N_\mathrm{s} - 1$ starting from the initial value $H_0^\mathrm{S} = \langle \lambda_0^x, \mathrm{D}_p^2 x_0(p) \rangle$. This symmetric variable satisfies $H_n^\mathrm{S} = \langle \lambda_n^x, \mathrm{D}_p^2 x_n \rangle \in \mathbb{R}^{n_p \times n_p}$ and yields the desired Hessian result $H_{N_\mathrm{s}}^\mathrm{S} = \langle \bar{\lambda}, \mathrm{D}_p^2 x_{N_\mathrm{s}} \rangle$.

Proof. The proof uses induction over n. For the case $n = 0$, the statement $H_0^\mathrm{S} = \langle \lambda_0^x, \mathrm{D}_p^2 x_0 \rangle$ for the symmetric Hessian result holds by initialization. Let us now assume that $H_n^\mathrm{S} = \langle \lambda_n^x, \mathrm{D}_p^2 x_n \rangle$ holds for n. From the symmetric sequence in Eq. (3.16), the following then holds for the case $n + 1$:

$$\begin{aligned} H_{n+1}^\mathrm{S} &= \langle \lambda_n^x, \mathrm{D}_p^2 x_n \rangle \\ &\quad + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle + \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle + \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle + \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle + \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}. \end{aligned} \tag{3.17}$$

From this expression, we can prove the desired result $H_{n+1}^\mathrm{S} = \langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1} \rangle$ based on the second order chain rule and the implicit function theorem. Using equation $x_{n+1} = F(x_n, z_n)$ from (3.10), this second order chain rule reads:

$$\begin{aligned} \langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1} \rangle &= \langle \tilde{\lambda}_n^x, \mathrm{D}_p^2 x_n \rangle + \langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle \\ &\quad + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n \rangle & \langle \lambda_{n+1}^x, F_{xz}^n \rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n \rangle & \langle \lambda_{n+1}^x, F_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \end{aligned} \tag{3.18}$$

where auxiliary variables $\tilde{\lambda}_n^{x^\top} := \lambda_{n+1}^{x^\top} F_x^n$ and $\tilde{\lambda}_n^{z^\top} := \lambda_{n+1}^{x^\top} F_z^n$ are defined. Let us recall the adjoint propagation from (3.13), where $0 = {F_z^n}^\top \lambda_{n+1}^x + {G_z^n}^\top \lambda_{n+1}^z$ and therefore $\lambda_{n+1}^{z^\top} = -\lambda_{n+1}^{x^\top} F_z^n {G_z^n}^{-1} = -\tilde{\lambda}_n^{z^\top} {G_z^n}^{-1}$ holds. The implicit function theorem for the equation $0 = G(x_n, z_n)$ then allows us to write the following directional second order derivatives as:

$$\langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle = \langle \tilde{\mu}_n^x, \mathrm{D}_p^2 x_n \rangle + \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^z, G_{xx}^n \rangle & \langle \lambda_{n+1}^z, G_{xz}^n \rangle \\ \langle \lambda_{n+1}^z, G_{zx}^n \rangle & \langle \lambda_{n+1}^z, G_{zz}^n \rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \tag{3.19}$$

where additionally $\tilde{\mu}_n^{x^\top} := \lambda_{n+1}^{z^\top} G_x^n$ is defined and $\lambda_{n+1}^{z^\top} G_z^n = -\tilde{\lambda}_n^{z^\top}$ has been used. After combining the expression for $\langle \tilde{\lambda}_n^z, \mathrm{D}_p^2 z_n \rangle$ in (3.19) into the result

from Eq. (3.18), one obtains:

$$\begin{aligned}\langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1}\rangle &= \langle \tilde{\lambda}_n^x, \mathrm{D}_p^2 x_n\rangle + \langle \tilde{\mu}_n^x, \mathrm{D}_p^2 x_n\rangle \\ &+ \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n\rangle + \langle \lambda_{n+1}^z, G_{xx}^n\rangle & \langle \lambda_{n+1}^x, F_{xz}^n\rangle + \langle \lambda_{n+1}^z, G_{xz}^n\rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n\rangle + \langle \lambda_{n+1}^z, G_{zx}^n\rangle & \langle \lambda_{n+1}^x, F_{zz}^n\rangle + \langle \lambda_{n+1}^z, G_{zz}^n\rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix} \\ &= \langle \lambda_n^x, \mathrm{D}_p^2 x_n\rangle \\ &+ \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}^\top \begin{bmatrix} \langle \lambda_{n+1}^x, F_{xx}^n\rangle + \langle \lambda_{n+1}^z, G_{xx}^n\rangle & \langle \lambda_{n+1}^x, F_{xz}^n\rangle + \langle \lambda_{n+1}^z, G_{xz}^n\rangle \\ \langle \lambda_{n+1}^x, F_{zx}^n\rangle + \langle \lambda_{n+1}^z, G_{zx}^n\rangle & \langle \lambda_{n+1}^x, F_{zz}^n\rangle + \langle \lambda_{n+1}^z, G_{zz}^n\rangle \end{bmatrix} \begin{bmatrix} S_n^x \\ S_n^z \end{bmatrix}, \end{aligned} \tag{3.20}$$

where we used that $\lambda_n^x = {F_x^n}^\top \lambda_{n+1}^x + {G_x^n}^\top \lambda_{n+1}^z = \tilde{\lambda}_n^x + \tilde{\mu}_n^x$ from Eq. (3.13). This concludes the induction proof, because (3.20) shows that $H_{n+1}^{\mathrm{S}} = \langle \lambda_{n+1}^x, \mathrm{D}_p^2 x_{n+1}\rangle$ holds, based on the original expression in Eq. (3.17). □

Let us briefly compare the classical FOA sensitivity propagation in Eq. (3.15) with the symmetric scheme in Eq. (3.16). One can observe that the novel equations propagate much less variables $H_n^{\mathrm{S}} \in \mathbb{R}^{n_p \times n_p}$ in case $n_p \ll (n_x + n_z)$, while the FOA variables are $H_n^x \in \mathbb{R}^{n_x \times n_p}$ and $H_n^z \in \mathbb{R}^{n_z \times n_p}$. In addition, the sensitivity equation (3.16) is symmetric, such that one can propagate only the lower triangular part of the variable $H_n^{\mathrm{S}} \in \mathbb{R}^{n_p \times n_p}$. Since this symmetric equation does not directly depend on H_n^{S} itself, the Hessian variable can be propagated in any direction. In Section 3.4, a three-sweep propagation scheme will be proposed in which these properties are exploited.

3.3 Continuous-Time Sensitivity Propagation

This section presents continuous-time sensitivity equations to propagate the first and second order directional derivatives with respect to the parameter p. This follows a differentiate-then-discretize type of approach as discussed in Section 2.3.3 of the previous chapter. We introduce the shorthand $f(t) = f(x(t), z(t))$ and $g(t) = g(x(t), z(t))$ for the DAE system in Eq. (3.3) whenever it is clear from the context at which point f and g are evaluated. Additionally, the partial derivatives of these functions are denoted by

$$f_x(t) = \partial_x f(t),\ f_z(t) = \partial_z f(t) \quad \text{and} \quad g_x(t) = \partial_x g(t),\ g_z(t) = \partial_z g(t).$$

The dependence of the simulation result $x(T, p)$ and the function $x_0(p)$ on the parameter value p is further omitted to allow a more compact notation.

Remark 3.5 *The continuous-time sensitivity equations could be derived from the discrete-time results in Section 3.2, by applying the limit for the discretization step size going to zero. For completeness, we however present these continuous-time results and instead provide a self-contained proof of correctness for the proposed symmetric Hessian propagation scheme.*

3.3.1 First Order Sensitivity Equations

Forward propagation Let us define the sensitivities $S^x(t) := \mathrm{D}_p x(t) \in \mathbb{R}^{n_x \times n_p}$ and $S^z(t) := \mathrm{D}_p z(t) \in \mathbb{R}^{n_z \times n_p}$. The forward system of sensitivity equations [67, 106] corresponding to the DAE in (3.3) can be written as:

$$\begin{aligned} \dot{S}^x(t) &= f_x(t) S^x(t) + f_z(t) S^z(t), \quad \text{with} \quad S^x(0) = \mathrm{D}_p x_0 \\ 0 &= g_x(t) S^x(t) + g_z(t) S^z(t), \end{aligned} \tag{3.21}$$

which is also of index 0 or 1 under Assumption 3.2. The end value $S^x(T)$ can be used to obtain the gradient result $\mathrm{D}_p \mathcal{M}(p)^\top = \mathrm{D}_p(x_T)^\top \bar{\lambda} = S^x(T)^\top \bar{\lambda}$ in the Newton-type optimization scheme.

Adjoint propagation The sensitivity result $\mathrm{D}_p \mathcal{M}(p)^\top$ can alternatively be computed directly by use of an adjoint propagation scheme. For this purpose, we define the adjoint variables $\lambda^x(t) := \mathrm{D}_{x(t)} x(T)^\top \bar{\lambda} \in \mathbb{R}^{n_x}$ and $\lambda^z(t) \in \mathbb{R}^{n_z}$. As derived in detail by [65, 66], the adjoint system then reads as:

$$\begin{aligned} -\dot{\lambda}^x(t) &= f_x(t)^\top \lambda^x(t) + g_x(t)^\top \lambda^z(t) \\ 0 &= f_z(t)^\top \lambda^x(t) + g_z(t)^\top \lambda^z(t), \end{aligned} \tag{3.22}$$

which is again of index 0 or 1 and the initial value $\lambda^x(T) = \bar{\lambda}$ is the backward seed. The adjoint result $\mathrm{D}_p(x_T)^\top \bar{\lambda} = \mathrm{D}_p(x_0)^\top \lambda^x(0)$ is then obtained directly by this backward sensitivity propagation.

3.3.2 Second Order Sensitivity Equations

Now, we are interested in computing second order directional derivatives of the form $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$ as required in Eq. (3.9) and following our notation in Eq. (3.1). Let us start by presenting the classical *forward-over-adjoint* (FOA) type sensitivity equations in continuous-time.

Forward-over-adjoint (FOA) Applying forward differentiation directly to the adjoint system (3.22) yields $H^x(t) := \mathrm{D}_p\lambda^x(t) \in \mathbb{R}^{n_x \times n_p}$ and $H^z(t) := \mathrm{D}_p\lambda^z(t) \in \mathbb{R}^{n_z \times n_p}$. These variables are described by the *FOA system* [240]:

$$\begin{aligned} -\dot{H}^x(t) &= f_x(t)^\top H^x(t) + g_x(t)^\top H^z(t) \\ &\quad + \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix} \\ 0 &= f_z(t)^\top H^x(t) + g_z(t)^\top H^z(t) \\ &\quad + \begin{bmatrix} \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \end{aligned} \tag{3.23}$$

where the first order sensitivities are defined earlier and the second order derivatives read $f_{a\,b} = \partial^2_{a,b} f(t)$ and $g_{a\,b} = \partial^2_{a,b} g(t)$. The initial value needs to be chosen $H^x(T) = \mathbb{0}$ and consistent values for $H^z(T)$ can be obtained similar to Definition 3.3. The Hessian result can then be evaluated as $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle = \mathrm{D}_p(x_0)^\top H^x(0) + \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. It is interesting to note that the sensitivity equation (3.23) is not symmetric, unlike the resulting Hessian.

3.3.3 Symmetric Second Order Sensitivity Equations

Instead of the FOA system in Eq. (3.23), we would like to directly propagate a symmetric variable $H^{\mathrm{S}}(t) \in \mathbb{R}^{n_p \times n_p}$. We show that both approaches can provide the same second order sensitivities $\langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$.

Theorem 3.6 (Symmetric sensitivity equations) *Let $(x(t), z(t))$, $(S^x(t), S^z(t))$ and $(\lambda^x(t), \lambda^z(t))$ be defined for $t \in [0, T]$ respectively by Eqs.* (3.3), (3.21) *and* (3.22) *and the corresponding consistent initial values. Let us introduce the following symmetric sensitivity equations to propagate $H^{\mathrm{S}}(t)$:*

$$\dot{H}^{\mathrm{S}}(t) = \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \tag{3.24}$$

for $t \in [0, T]$, starting from the initial value $H^{\mathrm{S}}(0) = \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. The symmetric variable satisfies $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ and provides the desired Hessian result $H^{\mathrm{S}}(T) = \langle \bar{\lambda}, \mathrm{D}_p^2 x_T \rangle$.

Proof. The expression $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ is satisfied for $t = 0$ because of the initialization $H^{\mathrm{S}}(0) = \langle \lambda^x(0), \mathrm{D}_p^2 x_0(p) \rangle$. It is sufficient to show that the time derivative of this expression satisfies the differential equation system (3.24),

in order to obtain the desired Hessian result. We start by differentiating $H^{\mathrm{S}}(t) = \langle \lambda^x(t), \mathrm{D}_p^2 x(t) \rangle$ with respect to time:

$$\dot{H}^{\mathrm{S}}(t) = \langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + \langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle. \tag{3.25}$$

The second order chain rule for the derivative $\langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle$ reads as:

$$\langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle = -\langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t), \tag{3.26}$$

where $\lambda^x(t)^\top \mathrm{D}_{x(t)} \dot{x}(t) = -\dot{\lambda}^x(t)^\top$ has been used and the directional derivative is defined as $\langle \lambda^x, \mathrm{D}_p^2 \dot{x} \rangle = \sum_{k=1}^{n_x} \lambda_k^x \, \mathrm{D}_p^2 \dot{x}_k$ following our notation in Eq. (3.1). The expression (3.26) can be used to simplify Eq. (3.25):

$$\begin{aligned} \dot{H}^{\mathrm{S}}(t) &= \langle \dot{\lambda}^x(t), \mathrm{D}_p^2 x(t) \rangle + \langle \lambda^x(t), \mathrm{D}_p^2 \dot{x}(t) \rangle \\ &= S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t). \end{aligned} \tag{3.27}$$

By differentiating the differential equation $\dot{x}(t) = f(x(t), z(t))$ twice with respect to x, we can write the directional second order derivative:

$$\begin{aligned} \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle = &\begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix} \\ &+ \langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle, \end{aligned} \tag{3.28}$$

where $\tilde{\lambda}^z(t)^\top := \lambda^x(t)^\top f_z(t)$ has been defined. Let us derive the remaining directional derivative $\langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle$ by also differentiating the algebraic equation $0 = g(x(t), z(t))$ twice with respect to x:

$$0 = \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} \mathbb{1} \\ \mathrm{D}_{x(t)} z(t) \end{bmatrix} - \langle \tilde{\lambda}^z(t), \mathrm{D}_x^2 z(t) \rangle, \tag{3.29}$$

where we used that $\tilde{\lambda}^z(t)^\top = \lambda^x(t)^\top f_z(t) = -\lambda^z(t)^\top g_z(t)$, based on the second expression in the adjoint system (3.22). Combining the expressions from Eqs. (3.29) and (3.28) into Eq. (3.27), yields the differential equation:

$$\begin{aligned} \dot{H}^{\mathrm{S}}(t) &= S^x(t)^\top \langle \lambda^x(t), \mathrm{D}_x^2 \dot{x}(t) \rangle S^x(t) \\ &= \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}^\top \begin{bmatrix} \langle \lambda^x, f_{x\,x} \rangle + \langle \lambda^z, g_{x\,x} \rangle & \langle \lambda^x, f_{x\,z} \rangle + \langle \lambda^z, g_{x\,z} \rangle \\ \langle \lambda^x, f_{z\,x} \rangle + \langle \lambda^z, g_{z\,x} \rangle & \langle \lambda^x, f_{z\,z} \rangle + \langle \lambda^z, g_{z\,z} \rangle \end{bmatrix} \begin{bmatrix} S^x(t) \\ S^z(t) \end{bmatrix}, \end{aligned} \tag{3.30}$$

where we used $S^z(t) = \mathrm{D}_{x(t)} z(t)\, S^x(t)$ and which corresponds to the desired symmetric sensitivity equation (3.24), concluding our proof. □

The symmetry of System (3.24) can be exploited by propagating its lower triangular part only. Notice also that unlike the classical FOA approach, the symmetric scheme results in an ODE system describing the propagation of the Hessian result $H^{\mathrm{S}}(t)$, although the original system is in general an implicit DAE system of index 1. In addition, the time direction in which these equations are simulated is arbitrary and can therefore be reversed. Note that a rather similar theorem and proof could therefore be constructed for a backward in time propagation of the symmetric system.

3.4 Three-Sweep Hessian Propagation Scheme

A stored trajectory for the state variables $x(t), z(t)$ from (3.3) for $t \in [0, T]$ is needed to simulate the adjoint equations and similar requirements hold for the second order sensitivity equations. This section presents an alternative approach for second order sensitivity propagation, based on the symmetric right hand of Eq. (3.24). Even though the algorithm description will be based on the continuous-time sensitivity equations from Section 3.3, the same techniques can be applied to the discrete-time propagation schemes from Section 3.2 based on the symmetric matrix equations in (3.16).

The classical approach for second order sensitivity propagation performs a forward sweep, followed by a backward sweep. We will therefore refer to this as the forward-backward (FB) propagation technique. As illustrated in Algorithm 1, the trajectories of $x(t), z(t)$ and $S^x(t), S^z(t)$ from the forward simulation over $t \in [0, T]$ need to be stored. In the following backward sweep, the adjoint derivatives $\lambda^x(t), \lambda^z(t)$ as well as the second order derivatives are propagated. The latter can be based either on the FOA equations from (3.23) using $H^x(t), H^z(t)$ or based on the proposed symmetric scheme in (3.24) which propagates directly the symmetric variable $H^{\mathrm{S}}(t)$.

Algorithm 1: Forward-backward Hessian propagation

Input: The initial value $x_0(p)$.

$\longrightarrow$ Propagate and store $x(t), z(t)$ in (3.3) and $S^x(t), S^z(t)$ in (3.21).

Evaluate backward seed $\bar{\lambda}^\top = \partial_x m\left(x(T,p)\right)$ in Eq. (3.9).

$\longleftarrow$ Propagate backward $\lambda^x(t), \lambda^z(t)$ in Eq. (3.22) and $H^x(t), H^z(t)$ in Eq. (3.23) or $H^{\mathrm{S}}(t)$ in Eq. (3.24).

Output: The results for $x(T,p)$, $\mathrm{D}_p x(T,p)^\top \bar{\lambda}$ and $\langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$.

As mentioned earlier, the symmetric equations represent a plain summation independent of the current value of $H^{\mathrm{S}}(\cdot)$ such that the time direction in which they are simulated can be reversed. This yields an alternative to the above forward-backward propagation, which consists of three consecutive sweeps. Algorithm 2 illustrates this three-sweep propagation (TSP) technique [268]. Note that it can be computationally better to perform less sweeps in order to allow more reuse of expressions in the derivative evaluations [141], even though the TSP will show other advantages over the FB propagation. The main advantage is that storage of the full forward trajectory of $S^x(t), S^z(t)$ can be avoided, since these first order derivatives are propagated together with the symmetric Hessian results in the third sweep.

Algorithm 2: Three-sweep Hessian propagation (TSP) for symmetric scheme

Input: The initial value $x_0(p)$.

$\longrightarrow$ Propagate forward and store $x(t), z(t)$ in Eq. (3.3).

Evaluate backward seed $\bar{\lambda}^\top = \partial_x m\left(x(T,p)\right)$ in Eq. (3.9).

$\longleftarrow$ Propagate backward and store $\lambda^x(t), \lambda^z(t)$ in Eq. (3.22).

$\longrightarrow$ Propagate forward $S^x(t), S^z(t)$ in Eq. (3.21) and $H^{\mathrm{S}}(t)$ in Eq. (3.24).

Output: The results for $x(T,p)$, $\mathrm{D}_p x(T,p)^\top \bar{\lambda}$ and $\langle \bar{\lambda}, \mathrm{D}_p^2 x(T,p) \rangle$.

3.4.1 Implementation Details: TSP versus FB Scheme

Table 3.1 shows a comparison between the FOA and the symmetric sensitivity equations. Note that the table presents the dimensions of the propagated matrix variables in both schemes, which does not directly correspond to the computational complexity. The latter depends on the efficient evaluation of the derivatives, on which more information can be found in [141]. In summary, the symmetric Hessian propagation from Eq. (3.24) in continuous time or Eq. (3.16) in discrete time provides the following advantages over the classical FOA scheme respectively from Eqs. (3.23) or (3.15):

- The matrix valued right hand in Eq. (3.24) or (3.16) is symmetric, i.e., only the lower or upper triangular part needs to be propagated. In the case where sensitivities are needed with respect to all states, which is common in direct optimal control, Table 3.1 shows that $n_x(n_x+1)/2$ equations need to be propagated instead of n_x^2. This could result in a speedup factor of about 2, depending on the sparsity of the problem.

Table 3.1: Theoretical cost comparison of second order sensitivity propagation.

Discrete-time	D-FOA eqs. (3.15)	D-SYM eqs. (3.16)
dimension (#variables)	$S_n^x, S_n^z, H_n^x, H_n^z$: $2\,n_x n_p + 2\,n_z n_p$	$S_n^x, S_n^z, H_n^{\mathrm{S}}$: $n_x n_p + n_z n_p + n_p(n_p+1)/2$
Continuous-time	C-FOA eqs. (3.23)	C-SYM eqs. (3.24)
dimension (#variables)	$S^x(t), S^z(t), H^x(t), H^z(t)$: $2\,n_x n_p + 2\,n_z n_p$	$S^x(t), S^z(t), H^{\mathrm{S}}(t)$: $n_x n_p + n_z n_p + n_p(n_p+1)/2$

- In case of implicit systems, the symmetric scheme is explicit and needs no additional variables. This is in contrast to the FOA equations.
- The derivatives in (3.24) or (3.16) can be evaluated using a symmetric AD technique for factorable functions as discussed in [140, 268].
- Since the direction of propagation for the symmetric equations can be reversed, one can avoid the storage of a trajectory of forward sensitivities based on the TSP scheme. Unlike the technique of checkpointing [141], this additional sweep typically causes a negligible amount of extra computational effort in the case of explicit differential equations [268].

Table 3.2 illustrates the storage costs for the FB and the TSP scheme respectively from Algorithm 1 and 2. The table shows that the TSP scheme can considerably reduce the memory requirements by propagating the first order forward sensitivities together with the symmetric Hessian propagation in a separate third sweep. A detailed comparison of these differentiation schemes would need to include additional factors such as the used integration method, the evaluation of derivatives in the sensitivity equations and the sparsity of the system dynamics as discussed in [141].

The presented sensitivity propagation techniques rely on the ability to simulate a certain system of differential equations both forward and backward in time. If this is not the case, one could employ a time reversing transformation as discussed in [65]. An efficient implementation of first order techniques can be found, e.g., in [166], which employs checkpointing to reduce storage requirements for adjoint sensitivity analysis. This approach could be applied in a similar way to the second order propagation schemes in this chapter.

Table 3.2: Storage cost for the second order sensitivity propagation techniques.

Discrete-time	Forward-backward (D-FB)	Three-sweep (D-TSP)
trajectory storage	x_n, z_n, S_n^x, S_n^z: $(n_x + n_z)(1 + n_p)$	$x_n, z_n, \lambda_n^x, \lambda_n^z$: $2\,(n_x + n_z)$
Continuous-time	**Forward-backward (C-FB)**	**Three-sweep (C-TSP)**
trajectory storage	$x(t), z(t), S^x(t), S^z(t)$: $(n_x + n_z)(1 + n_p)$	$x(t), z(t), \lambda^x(t), \lambda^z(t)$: $2\,(n_x + n_z)$

3.4.2 The TSP Scheme for Implicit Equations

It should be noted that the advantages of the TSP scheme regarding its storage requirements can become less apparent or even disappear in the case of implicit integration methods. To illustrate this, let us look at the integration scheme in Eq. (3.10) and its first order sensitivity propagation in Eq. (3.12) and (3.13). A Newton-type method to solve the implicit equation $0 = G(x_n, z_n)$ in (3.10) requires a factorization of the Jacobian G_z^n, which is also needed for the sensitivity equations (3.12) and (3.13). One needs to either store these Jacobian factorizations in the first sweep of Algorithm 1 and 2, or one needs to recompute them. It is possible that the advantage of using the TSP scheme over the classical forward-backward propagation, regarding its storage costs, becomes relatively small in this case. It therefore depends on the specific implementation, in the case of sensitivity propagation for implicit differential equations or implicit integration methods, whether the FB or the TSP scheme should be used. Nonetheless, the proposed symmetric sensitivity equations can still be used within the forward-backward propagation for its computational advantages as listed before.

3.4.3 Open-Source Implementation: ACADO Toolkit

An efficient implementation of the presented first and second order sensitivity propagation techniques for general DAE systems as well as of the symmetric AD method [268], can be found as part of the open-source `ACADO` code generation software [176, 177]. Table 3.3 illustrates the presented options for a tailored Hessian propagation, combining the FOA or symmetric sensitivity equations, with a FB or a TSP scheme within a discrete- or a continuous-time framework. Following the approach of Internal Numerical Differentiation (IND) [48], the open-source implementation in the `ACADO Toolkit` is based on a discrete-time

Table 3.3: The proposed second order sensitivity propagation techniques.

	Discrete-time	Continuous-time
FB	D-FB-FOA (3.15) D-FB-SYM (3.16)	C-FB-FOA (3.23) C-FB-SYM (3.24)
TSP	D-TSP-SYM (3.16)	C-TSP-SYM (3.24)

Table 3.4: Computation times for a numerical simulation of the chain mass [325] using explicit RK4: TSP-SYM versus FB-FOA sensitivity propagation.

		D-TSP-SYM		D-FB-FOA	
n_{m}	n_x	$n_p = n_x + n_u$	$n_p = n_u$	$n_p = n_x + n_u$	$n_p = n_u$
3	12	133 µs	29 µs	326 µs	57 µs
4	18	376 µs	66 µs	979 µs	125 µs
5	24	826 µs	103 µs	1945 µs	188 µs
6	30	1523 µs	139 µs	3219 µs	245 µs
7	36	2495 µs	176 µs	4970 µs	300 µs
8	42	3710 µs	211 µs	6902 µs	386 µs
9	48	5293 µs	255 µs	9650 µs	461 µs

sensitivity propagation, i.e., corresponding to the schemes in the first column of Table 3.3. Note that the presented implementation targets mostly small to medium-scale systems of about 10-100 states [176, 271], even though this does not limit the applicability of the proposed algorithmic techniques to such problems. More information on this open-source software implementation and the `ACADO` code generation tool can be found in Chapter 8.

As an example, let us briefly look at the ODE model equations for a chain of n_{m} masses as described in [325]. Table 3.4 presents the timing results for a numerical simulation over 0.5 s using 10 steps of the explicit Runge-Kutta (RK) method of order 4, based on the D-TSP-SYM or the D-FB-FOA scheme. The table shows the computation times for a second order sensitivity analysis with respect to both the states and control variables ($n_p = n_x + n_u$) or only the controls ($n_p = n_u = 3$), using different numbers of masses n_{m} with the corresponding state dimension $n_x = 6\,(n_{\mathrm{m}} - 1)$. It can be observed from this table that the overall computational speedup is about factor 2 based on the symmetry of the proposed sensitivity equations in (3.16). The following section finally confirms these results on an illustrative case study of the economic optimal control of a nonlinear biochemical reactor.

3.5 Numerical Case Study

The numerical results in this section have been obtained using the open-source `ACADO` code generation tool on a standard computer, equipped with Intel i7-3720QM processor, and using a 64-bit version of Ubuntu 14.04 and the g++ compiler version 4.8.4. The code to reconstruct the presented numerical results can be found on the following public repository: `https://github.com/rienq/symmetricHessians`.

3.5.1 Modeling the Dynamic System

We consider the following explicit ODE model of a continuous bioreactor for culture fermentation [201, 244, 286]:

$$\dot{x} = f_{\text{reactor}}(x,u) = \begin{pmatrix} -DX_{\text{b}} + \mu(x)X_{\text{b}} \\ D(U_{\text{f}} - X_{\text{s}}) - \frac{\mu(x)X_{\text{b}}}{Y_{\text{b}}} \\ -DX_{\text{p}} + (\alpha\mu(x) + \beta)X_{\text{b}} \\ X_{\text{b}}/T \\ U_{\text{f}}/T \\ DX_{\text{p}}/T \end{pmatrix} . \tag{3.31}$$

Here, the state vector $x = (X_{\text{b}}, X_{\text{s}}, X_{\text{p}}, q_{\text{b}}, q_{\text{f}}, q_{\text{p}})$ consists of three physical states: the biomass X_{b}, the substrate X_{s} and the product concentration X_{p} as well as three auxiliary states q_{b}, q_{f} and q_{p}. The latter auxiliary variables are also known as quadrature states [166], e.g., $\dot{q}_{\text{b}}(t) = X_{\text{b}}(t)/T$ such that $q_{\text{b}}(T) = \frac{1}{T}\int_0^T X_{\text{b}}(\tau)\text{d}\tau$ denotes the average biomass concentration. These quadrature states will be used further to formulate the objective and constraint functions in the OCP. The control input $u = U_{\text{f}}$ of the system is the feed substrate concentration which is bounded $\underline{U} \leq U_{\text{f}} \leq \overline{U}$. The specific growth rate $\mu(x)$ is given by the expression

$$\mu(x) = \mu_{\text{m}} \frac{(1 - \frac{X_{\text{p}}}{P_{\text{m}}} X_{\text{s}})}{K_{\text{m}} + X_{\text{s}} + \frac{{X_{\text{s}}}^2}{K_{\text{i}}}} \ .$$

The remaining parameter values and operating bounds are summarized in Table 3.5. The bioreactor can be modeled by an explicit ODE as in (3.31), but we later also refer to the following semi-explicit DAE formulation:

$$\begin{aligned} \dot{x} &= f_{\text{reactor}}(x, \mu, u) \\ 0 &= \mu - \mu_{\text{m}} \frac{(1 - \frac{X_{\text{p}}}{P_{\text{m}}} X_{\text{s}})}{K_{\text{m}} + X_{\text{s}} + \frac{{X_{\text{s}}}^2}{K_{\text{i}}}}, \end{aligned} \tag{3.32}$$

Table 3.5: Parameter values and bounds for the bioreactor.

Name	Symbol	Value
dilution rate	D	$0.15\,\mathrm{h}^{-1}$
substrate inhibition constant	K_i	$22\,\mathrm{g/L}$
substrate saturation constant	K_m	$1.2\,\mathrm{g/L}$
product saturation constant	P_m	$50\,\mathrm{g/L}$
yield of the biomass	Y_b	0.4
first product yield constant	α	2.2
second product yield constant	β	$0.2\,\mathrm{h}^{-1}$
specific growth rate scale	μ_m	$0.48\,\mathrm{h}^{-1}$
maximum feed substrate	$\overline{U}$	$40.0\,\mathrm{g/L}$
minimum feed substrate	$\underline{U}$	$28.7\,\mathrm{g/L}$
maximum average feed substrate	$\overline{U}^\mathrm{A}$	$32.9\,\mathrm{g/L}$
maximum average biomass concentration	$\overline{X}_\mathrm{b}^\mathrm{A}$	$5.8\,\mathrm{g/L}$

where the specific growth rate has been introduced as an algebraic variable. Even though there is no clear advantage of using the DAE formulation in this case, it can be used to illustrate the proposed sensitivity propagation schemes for implicit systems of equations.

3.5.2 Optimal Control Problem Formulation

Similar to the OCP formulation in [268], our aim here is to maximize the average productivity which corresponds to the Mayer term $q_\mathrm{p}(T)$ as described by Eq. (3.31). The end time T of a single cycle is assumed to be fixed and equal

to 48 hours. The resulting continuous-time OCP formulation reads as:

$$\min_{x(\cdot),u(\cdot)} \quad -q_{\mathrm{p}}(T) \tag{3.33a}$$

$$\text{s.t.} \quad 0 = x(0) - \hat{x}_0, \tag{3.33b}$$

$$\dot{x}(t) = f_{\mathrm{reactor}}(x,u), \quad \forall t \in [0,T], \tag{3.33c}$$

$$\underline{U} \leq U_{\mathrm{f}} \leq \overline{U}, \tag{3.33d}$$

$$q_{\mathrm{f}}(T) \leq \overline{U}^{\mathrm{A}}, \tag{3.33e}$$

$$q_{\mathrm{b}}(T) \leq \overline{X}_{\mathrm{b}}^{\mathrm{A}}, \tag{3.33f}$$

$$X_{\mathrm{b}}(T) = X_{\mathrm{b}}(0), \tag{3.33g}$$

$$X_{\mathrm{s}}(T) = X_{\mathrm{s}}(0), \tag{3.33h}$$

$$X_{\mathrm{p}}(T) = X_{\mathrm{p}}(0), \tag{3.33i}$$

which, in addition to the initial value condition (3.33b) and the dynamics (3.33c), also includes the control bounds (3.33d) and an upper bound on the average concentration for feed substrate (3.33e) and biomass (3.33f). Following the problem formulation in [268, 286], periodicity constraints on the three physical states X_{b}, X_{s} and X_{p} are included in Eqs. (3.33g)-(3.33i). Note that the initial value $\hat{x}_0$ for the states is considered constant, unlike the parametric OCP formulation from Eq. (3.6). Instead, the sensitivity information with respect to the control inputs will be needed in a Newton-type optimization method for this OCP (3.33) after performing a shooting discretization [48]. For the sake of simplicity, we consider a piecewise constant control parameterization over the horizon of $N = 20$ equidistant intervals. The solution trajectories for the states and controls are shown in Figure 3.1.

3.5.3 Numerical Simulation Results

As discussed earlier in Section 3.4.3, the open-source `ACADO Toolkit` can be used to efficiently solve the OCP in Eq. (3.33), based on direct multiple shooting and an SQP type algorithm. The presented sensitivity propagation techniques will be illustrated in discrete-time, using 5 integration steps of the form in Eq. (3.10) within each of the $N = 20$ shooting intervals over the control horizon of $T = 48$ hours. Let us use the explicit Runge-Kutta (RK) method of order 4 for the ODE model in Eq. (3.31). Respectively, we use an implicit RK method

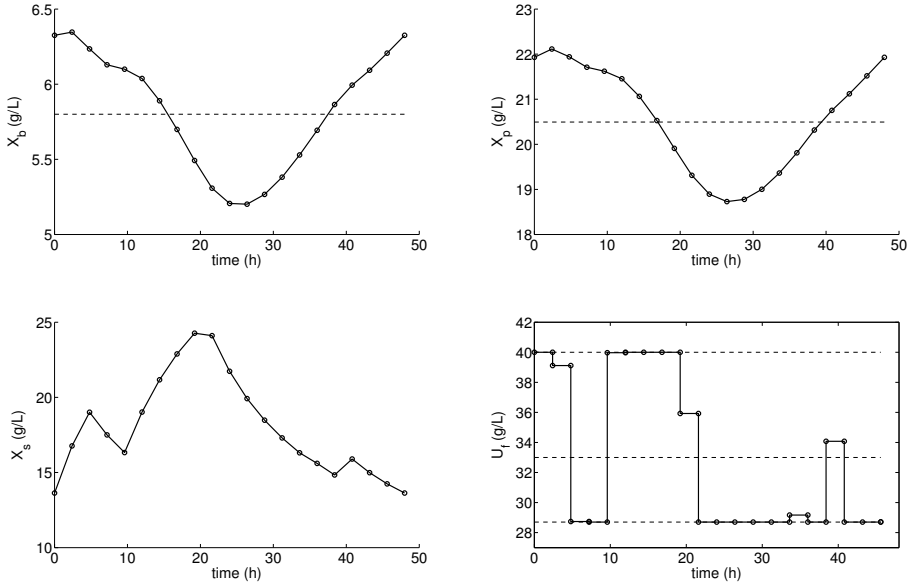


Figure 3.1: Illustration of the optimal state and control trajectories, corresponding to the periodic OCP for the nonlinear bioreactor in Eq. (3.33).

of order 4 to simulate the semi-explicit DAE formulation from Eq. (3.32). Note that this DAE system is equivalent to the original ODE model and has been introduced mainly to illustrate the sensitivity propagation techniques in the presence of implicit algebraic equations.

Table 3.6 details the average computation times for the different components during one iteration of the SQP algorithm, using the explicit RK4 method for the ODE model. The table includes a comparison of the proposed symmetric three-sweep Hessian propagation (TSP-SYM) versus the classical forward-over-adjoint based FB scheme (FB-FOA). The symmetric propagation method indeed yields a speedup factor of about 2 as discussed in Section 3.4.1, when comparing the total simulation time. Moreover, recall from Table 3.2 that the TSP scheme propagates all forward and second order sensitivities in a separate third sweep and is therefore also more memory efficient than the classical FB approach. Table 3.7 shows the average computation times using the implicit RK4 method for the DAE system, based on either the symmetric or the FOA equations for the forward-backward Hessian propagation. As mentioned earlier, the linear algebra routines for computing the Jacobian factorization often dominate the

Table 3.6: Detailed computation times for exact Hessian based SQP using the explicit RK4 method: TSP-SYM versus FB-FOA sensitivity propagation.

	D-TSP-SYM		D-FB-FOA	
Forward sweep 1	15 µs	6 %	95 µs	24 %
Backward sweep 2	18 µs	7 %	172 µs	44 %
Forward sweep 3	99 µs	39 %	-	-
Total simulation	135 µs	53 %	272 µs	69 %
Condensing	17 µs	7 %	19 µs	5 %
Regularization	47 µs	18 %	47 µs	12 %
QP solution	56 µs	22 %	56 µs	14 %
Total SQP step	256 µs		394 µs	

Table 3.7: Average computation times for exact Hessian based SQP using the implicit RK4 method: FB-SYM versus FB-FOA sensitivity propagation.

	D-FB-SYM		D-FB-FOA	
Total simulation	414 µs	74 %	451 µs	76 %
Total SQP step	557 µs		594 µs	

computational effort within an implicit scheme. This can also be observed in this table, since the total simulation times are relatively similar for both techniques here.

3.6 Conclusions and Outlook

In this chapter, we presented a novel sensitivity propagation technique to compute Hessian contributions as needed for Newton-type optimization in direct optimal control. By maintaining and exploiting the symmetry, we proposed a scheme which allows a computational speedup of about factor 2 over the classical forward-over-adjoint approach for both discrete- and continuous-time sensitivity analysis. The techniques have been presented in a general framework, including implicit integration methods and DAE systems of index up to 1. A novel three-sweep propagation technique has been proposed using this set of symmetric sensitivity equations, which results in a reduced memory footprint by avoiding the trajectory of forward sensitivities to be stored. Based on an

open-source implementation in the ACADO Toolkit, the performance has been illustrated for the economic optimal control of a nonlinear biochemical reactor.

The application of these symmetric Hessian propagation schemes for the solution of large-scale optimal control problems is part of ongoing research.

Chapter 4

Structure Exploitation for Linear Subsystems

An important algorithmic component in the implementation of Nonlinear MPC deals with the numerical simulation and propagation of sensitivities for the dynamic system. For this purpose, Chapter 2 presented an efficient implementation of fixed step collocation methods with a direct sensitivity analysis for embedded optimization. It is however known that the models used in practice, either ODE or index-1 DAE systems, often have a specific structure resulting in a sparse Jacobian matrix. A scalable way to handle this is by using general-purpose sparse linear algebra routines, which typically create little to no gain for small or even medium scale systems in real-time optimal control applications [121, 123]. The aim here is therefore to have a closer look at the structure specific to these dynamic systems and to propose an alternative approach for structure exploitation, that will be shown to perform rather well. The addressed model class contains as a subclass the Wiener-Hammerstein variants which are since a long time used in system identification [124].

Note that this chapter is largely based on the article in [260].

Outline The chapter is organized as follows. In Section 4.1, the three stage model formulation is proposed. Section 4.2 then discusses the tailored structure exploitation within the collocation methods with direct sensitivity generation. Some real-world applications are used in Section 4.3 to illustrate the structure exploiting integrators and their numerical performance.

4.1 A Three-Stage Dynamic Structure

Of particular interest here is the nonlinear function $f(\cdot)$ in the continuous-time optimal control formulation from Eq. (1.6), which defines the system dynamics. This section proposes a particular structure, consisting of three subsequent stages in the system of differential equations.

4.1.1 Structured Model Formulation

When modeling, for example, a mechatronic system, the result is typically a set of nonlinear differential equations with possibly some algebraic variables. In the case of an explicit ODE such as in (1.2), one might recognize the following three subsystems in this specific order

$$\dot{x}^{[1]} = A_1 x^{[1]} + B_1 u, \qquad \text{(linear input system)} \tag{4.1a}$$

$$\dot{x}^{[2]} = f_2(x^{[1]}, x^{[2]}, u), \qquad \text{(nonlinear system)} \tag{4.1b}$$

$$\dot{x}^{[3]} = A_3 x^{[3]} + f_3(x^{[1]}, x^{[2]}, u), \qquad \text{(linear output system)} \tag{4.1c}$$

with the matrices A_1, B_1 and A_3 and the nonlinear functions $f_2(\cdot)$ and $f_3(\cdot)$ defining the subsystems. Figure 4.1 illustrates the proposed chain of subsystems. In what follows, $n_{\mathrm{x}}^{[1]}$, $n_{\mathrm{x}}^{[2]}$ and $n_{\mathrm{x}}^{[3]}$ denote the number of differential states in each of the three subsystems, i.e., $x^{[1]}(t) \in \mathbb{R}^{n_{\mathrm{x}}^{[1]}}$, $x^{[2]}(t) \in \mathbb{R}^{n_{\mathrm{x}}^{[2]}}$, $x^{[3]}(t) \in \mathbb{R}^{n_{\mathrm{x}}^{[3]}}$ and $n_{\mathrm{x}} = n_{\mathrm{x}}^{[1]} + n_{\mathrm{x}}^{[2]} + n_{\mathrm{x}}^{[3]}$. This system structure is inspired by Wiener-Hammerstein models [124], which consist of a linear dynamic system followed by a static nonlinearity and another linear dynamic system. In our notation, they would be of the form in (4.1) with $n_{\mathrm{x}}^{[2]} = 0$ in order to exclude nonlinear dynamics. Let us generalize this three-stage model structure even further to an implicit DAE system of index 1:

$$C_1 \dot{x}^{[1]} = A_1 x^{[1]} + B_1 u, \tag{4.2a}$$

$$0 = f_2(x^{[1]}, x^{[2]}, \dot{x}^{[1]}, \dot{x}^{[2]}, z, u), \tag{4.2b}$$

$$C_3 \dot{x}^{[3]} = A_3 x^{[3]} + f_3(x^{[1]}, x^{[2]}, \dot{x}^{[1]}, \dot{x}^{[2]}, z, u), \tag{4.2c}$$

with invertible matrices C_1, C_3 and invertible Jacobian $\frac{\partial f_2}{\partial z, \dot{x}^{[2]}}(\cdot)$.

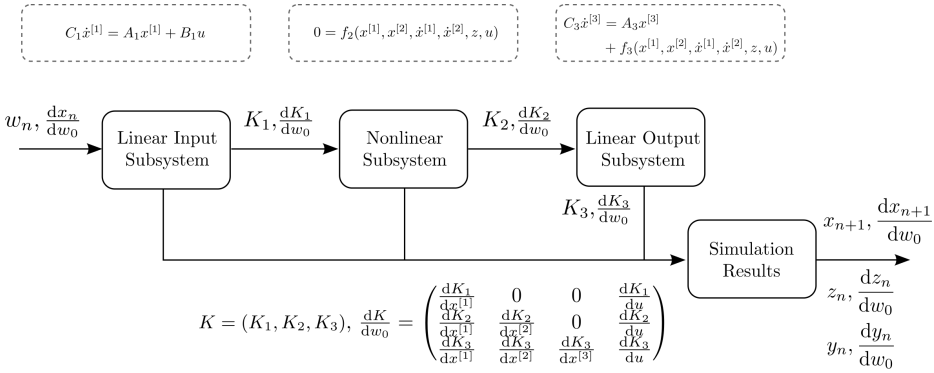


Figure 4.1: Schematic of the three-stage dynamic system structure, illustrating the workflow in the structure exploiting collocation based integrators.

4.1.2 Motivating Examples

The proposed structure is not something that comes up only in rare occasions. These additional stages often arise naturally when modeling for control. For example, a linear input system (4.2a) could result from

- partially linear dynamics which are independent of the nonlinear states. A classical example would be the double integrator

$$\begin{aligned} \dot{x}_1 &= u \\ \dot{x}_2 &= x_1 \end{aligned} \tag{4.3}$$

  but any linear set of equations is possible.

- any linear high order differential equation, which would be transformed into a set of first order equations

$$\begin{aligned} &\frac{\mathrm{d}^n x}{\mathrm{d}t^n} = h(t, u, x, \frac{\mathrm{d}x}{\mathrm{d}t}, \ldots, \frac{\mathrm{d}^{n-1}x}{\mathrm{d}t^{n-1}}) \\ &\qquad \Downarrow \quad x_1 = x, x_2 = \frac{\mathrm{d}x}{\mathrm{d}t}, \ldots, x_n = \frac{\mathrm{d}^{n-1}x}{\mathrm{d}t^{n-1}} \\ &\begin{cases} \dot{x}_1 &= x_2 \\ &\vdots \\ \dot{x}_n &= h(t, u, x_1, x_2, \ldots, x_n) \end{cases} \end{aligned} \tag{4.4}$$

  where $h(\cdot)$ denotes a linear function in this case.

- implementing a filter on the controls or input states. In some optimal control applications, it can be desirable to particularly penalize, e.g., the high frequency content. For this purpose, a high-pass RC filter

$$\frac{\mathrm{d}u^{\mathrm{HP}}}{\mathrm{d}t}(t) = \frac{\mathrm{d}u}{\mathrm{d}t}(t) - \omega_{\mathrm{c}} u^{\mathrm{HP}}(t) \tag{4.5}$$

  can be used, where $\omega_{\mathrm{c}} = \frac{1}{RC} = 2\pi f_{\mathrm{c}}$ in which f_{c} is typically called the *cutoff frequency*. Eq. (4.5) describes the additional state $u^{\mathrm{HP}}(t)$, which denotes the high-pass filtered result for the input $u(t)$.

The linear output system (4.2c) is somewhat similar to the input system, but with a possibly nonlinear relation with respect to the previous state variables and control inputs. An output system can therefore result from some partially linear dynamics that also depend on the previous states, such as, e.g., from introducing additional filter dynamics. When $A_3 = \mathbb{0}$ and $C_3 = \mathbb{1}$ is an identity matrix, Eq. (4.2c) reduces to

$$\dot{x}^{[3]} = f_3(x^{[1]}, x^{[2]}, \dot{x}^{[1]}, \dot{x}^{[2]}, z, u), \tag{4.6}$$

which are also known as quadrature states [166]. They are typically used to formulate objective and constraint functions in an OCP, similar to the more general in- and output subsystem states which we propose here.

4.2 Tailored Structure Exploiting IRK methods

This section presents how to adapt the IRK method from Eq. (2.28) to our proposed three-stage model structure in (4.2), including a detailed algorithm description and our open-source software implementation.

4.2.1 Three-Stage Structure Exploitation

The methods from Section 2.2, could directly be applied to the system in (4.2) without taking the specific structure into account. We aim however at an implementation which is mathematically equivalent with the latter, while exploiting the three-stage structure computationally. As illustrated in Figure 4.1, the scheme can be represented by four separate blocks including the one that handles the simulation results. Note that the latter could implement the continuous output feature as described in Section 2.5. The idea is to compute all collocation variables $K = (k_1, \ldots, k_q, Z_1, \ldots, Z_q)$ and their derivatives $\frac{\mathrm{d}K}{\mathrm{d}w_0}$ sequentially for the three model stages, where $w_n := (x_n, u)$ denotes the input

to the n^{th} integration step of the IRK scheme. The outputs and sensitivities can then be propagated in a similar way as before.

To be able to refer to the collocation variables within each stage separately, let us introduce the notation $K_1 = (k_1^{[1]}, \dots, k_q^{[1]})$, $K_2 = (k_1^{[2]}, \dots, k_q^{[2]}, Z_1, \dots, Z_q)$ and $K_3 = (k_1^{[3]}, \dots, k_q^{[3]})$ when applying the IRK method from (2.28) to the three-stage model structure in (4.2). This allows us to write the nonlinear system in a correspondingly structured way as follows

$$C_1 k_i^{[1]} = A_1(x_n^{[1]} + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j^{[1]}) + B_1 u, \tag{4.7a}$$

$$0 = f_2 \left(x_{n,i}^{[1]}, x_n^{[2]} + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j^{[2]}, k_i^{[1]}, k_i^{[2]}, Z_i, u \right), \tag{4.7b}$$

$$C_3 k_i^{[3]} = A_3(x_n^{[3]} + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j^{[3]}) + f_3(x_{n,i}^{[1]}, x_{n,i}^{[2]}, k_i^{[1]}, k_i^{[2]}, Z_i, u), \tag{4.7c}$$

for $i = 1, \dots, q$ and where the stage values $x_{n,i}^{[s]} = x_n^{[s]} + T_{\text{int}} \sum_{j=1}^{q} a_{ij} k_j^{[s]}$ are defined for $s = 1, 2, 3$. Based on the compact notation $0 = G(w_n, K)$ as introduced in (2.41), the nonlinear system can alternatively be written as

$$0 = G_1(w_n, K_1), \tag{4.8a}$$

$$0 = G_2(w_n, K_1, K_2), \tag{4.8b}$$

$$0 = G_3(w_n, K_1, K_2, K_3), \tag{4.8c}$$

where the next state value reads as $x_{n+1} = F(w_n, K)$.

Linear input system The collocation variables K_1 are defined by the linear equations in (4.7a), which could also be more compactly written as in (4.8a). The corresponding Jacobian

$$M_1 = \frac{\partial G_1}{\partial K_1} = \begin{pmatrix} C_1 - T_{\text{int}} a_{11} A_1 & \cdots & -T_{\text{int}} a_{1q} A_1 \\ \vdots & \ddots & \vdots \\ -T_{\text{int}} a_{q1} A_1 & \cdots & C_1 - T_{\text{int}} a_{qq} A_1 \end{pmatrix} \tag{4.9}$$

is therefore a constant matrix. Instead of performing an iterative solution, the collocation variables can be obtained directly by solving the linear system of equations, i.e., $K_1 = -M_1^{-1} G_1(w_n, 0)$. The inverse matrix M_1^{-1} remains

constant for each fixed step integration and could be computed beforehand. In addition to the evaluation of the linear function $G_1(\cdot)$, the linear algebra then reduces to a matrix-vector multiplication. Similarly, the first order derivatives $\frac{\mathrm{d}K_1}{\mathrm{d}w_n}$ are constant and can be precomputed. The forward sensitivity propagation then corresponds to the expression $\frac{\mathrm{d}K_1}{\mathrm{d}w_0} = \frac{\mathrm{d}K_1}{\mathrm{d}w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0}$.

Nonlinear system For the implicit nonlinear system in (4.7b) or (4.8b), let us use the iterative implementation with a direct sensitivity propagation as discussed in Section 2.4 and detailed in the IFT-R scheme of Algorithm 2. The main difference here is that the set of nonlinear equations depends on the collocation variables K_1 from the linear input subsystem. For this reason, the derivatives $\frac{\mathrm{d}K_1}{\mathrm{d}w_0}$ are needed to propagate the first order sensitivities

$$\frac{\mathrm{d}K_2}{\mathrm{d}w_0} = -\frac{\partial G_2}{\partial K_2}^{-1}\left(\frac{\partial G_2}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial G_2}{\partial K_1}\frac{\mathrm{d}K_1}{\mathrm{d}w_0}\right).$$

Linear output system Similar to the input subsystem, the collocation variables for the linear output system can be computed as $K_3 = -M_3^{-1}G_3(w_n, K_1, K_2, 0)$ in which $M_3 = \frac{\partial G_3}{\partial K_3}$ denotes the Jacobian. The inverse matrix M_3^{-1} and the derivatives $\frac{\mathrm{d}K_3}{\mathrm{d}x_n^{[3]}}$ are constant and could be precomputed in this case. The remaining first order derivatives however still need to be computed and can be propagated forward directly as follows

$$\frac{\mathrm{d}K_3}{\mathrm{d}w_0} = -M_3^{-1}\left(\frac{\partial G_3}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial G_3}{\partial K_1}\frac{\mathrm{d}K_1}{\mathrm{d}w_0} + \frac{\partial G_3}{\partial K_2}\frac{\mathrm{d}K_2}{\mathrm{d}w_0}\right).$$

4.2.2 Implementation Details

A description of the proposed collocation integrator with exploitation of the three-stage model structure can be found in Algorithm 3. It includes the optional continuous output feature which can be used to evaluate any (possibly nonlinear) function of the system variables on a certain evaluation grid. Note that an efficient implementation of Algorithm 3 takes into account that the Jacobian matrix $\frac{\mathrm{d}K}{\mathrm{d}w_0}$ has a clear sparsity structure

$$\frac{\mathrm{d}K}{\mathrm{d}w_0} = \begin{pmatrix} \frac{\mathrm{d}K_1}{\mathrm{d}x^{[1]}} & \mathbb{0} & \mathbb{0} & \frac{\mathrm{d}K_1}{\mathrm{d}u} \\ \frac{\mathrm{d}K_2}{\mathrm{d}x^{[1]}} & \frac{\mathrm{d}K_2}{\mathrm{d}x^{[2]}} & \mathbb{0} & \frac{\mathrm{d}K_2}{\mathrm{d}u} \\ \frac{\mathrm{d}K_3}{\mathrm{d}x^{[1]}} & \frac{\mathrm{d}K_3}{\mathrm{d}x^{[2]}} & \frac{\mathrm{d}K_3}{\mathrm{d}x^{[3]}} & \frac{\mathrm{d}K_3}{\mathrm{d}u} \end{pmatrix}. \tag{4.10}$$

In addition, it is interesting to note that the in- and output system actually have an analytical solution which is relatively easy to obtain since these subsystems

Algorithm 3 Collocation integration step: IFT-R with structure exploitation

Input: w_n, $\frac{\mathrm{d}x_n}{\mathrm{d}w_0}$, guess $K_2^{[0]}$, factorized M_2 and M_1^{-1}, M_3^{-1} and $\frac{\mathrm{d}K_1}{\mathrm{d}w_n}$.

`Linear input system` (4.7a) and (4.8a)

$K_1 \leftarrow -M_1^{-1} G_1(w_n, 0)$. ▷ M_1^{-1} is constant

$\frac{\mathrm{d}K_1}{\mathrm{d}w_0} \leftarrow \frac{\mathrm{d}K_1}{\mathrm{d}w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0}$. ▷ $\frac{\mathrm{d}K_1}{\mathrm{d}w_n}$ is constant

`Nonlinear system` (4.7b) and (4.8b)

for $i = 0 \rightarrow L-1$ **do**

  $K_2^{[i+1]} \leftarrow K_2^{[i]} - M_2^{-1} G_2(w_n, K_1, K_2^{[i]})$.

end for

$M_2 \leftarrow \frac{\partial G_2}{\partial K_2}(w_n, K_2^{[L]})$. ▷ factorization M_2

$\frac{\mathrm{d}K_2}{\mathrm{d}w_0} \leftarrow -M_2^{-1}\left(\frac{\partial G_2}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial G_2}{\partial K_1}\frac{\mathrm{d}K_1}{\mathrm{d}w_0}\right)$.

`Linear output system` (4.7c) and (4.8c)

$K_3 \leftarrow -M_3^{-1} G_3(w_n, K_1, K_2, 0)$. ▷ M_3^{-1} is constant

$\frac{\mathrm{d}K_3}{\mathrm{d}w_0} \leftarrow -M_3^{-1}\left(\frac{\partial G_3}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial G_3}{\partial K_1}\frac{\mathrm{d}K_1}{\mathrm{d}w_0} + \frac{\partial G_3}{\partial K_2}\frac{\mathrm{d}K_2}{\mathrm{d}w_0}\right)$. ▷ $\frac{\mathrm{d}K_3}{\mathrm{d}x_n^{[3]}}$ is constant

`Simulation results`

$x_{n+1} \leftarrow F(w_n, K)$.

$\frac{\mathrm{d}x_{n+1}}{\mathrm{d}w_0} \leftarrow \frac{\partial F}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial F}{\partial K}\frac{\mathrm{d}K}{\mathrm{d}w_0}$.

`Continuous output` (optional)

$y_n \leftarrow \Psi(w_n, K)$.

$\frac{\mathrm{d}y_n}{\mathrm{d}w_0} \leftarrow \frac{\partial \Psi}{\partial w_n}\frac{\mathrm{d}w_n}{\mathrm{d}w_0} + \frac{\partial \Psi}{\partial K}\frac{\mathrm{d}K}{\mathrm{d}w_0}$.

Output: x_{n+1}, $\frac{\mathrm{d}x_{n+1}}{\mathrm{d}w_0}$, next guess $K_2^{[0]}$ and factorized M_2.

consist of a set of linear differential equations with constant coefficients. Using this knowledge would generally however have little impact on the computational complexity and the overall accuracy of the numerical integration method for the complete nonlinear dynamic system.

An efficient implementation of the proposed integrators has been made part of the open-source `ACADO` code generation tool [176, 177], as presented also in [271] in the form of a tutorial. The code generation framework is developed in C++, which itself exports efficient C-code tailored to the specific problem formulation. A code example from `MATLAB` for the definition of linear subsystems in the dynamic model, for the `ACADO` code generation tool, can, e.g., be found in [260]. This way, the presented integrators can be automatically generated as standalone components or they can be exported directly as part of an OCP solver. The latter has been done in order to obtain the NMPC simulation results

of the next section. More information on these and other related open-source software developments can be found in Chapter 8.

Remark 4.1 *Even though we focus here on first order forward sensitivity propagation, it is clear that the proposed three-stage model structure could similarly result in computational advantages for an adjoint or second order sensitivity analysis. One could, for example, envision an efficient implementation of the symmetric Hessian propagation scheme from Chapter 3, which directly exploits the sparsity resulting from the linear subsystems. We further restrict to Gauss-Newton based optimal control, using first order forward sensitivity analysis as a relatively simple framework to illustrate the numerical performance of the proposed structure exploiting integrators.*

4.3 Optimal Control Application Examples

Let us now illustrate the numerical performance of the proposed structure exploiting integrators, based on two different case studies. For this purpose, we carry out closed-loop simulations of Nonlinear MPC using the Real-Time Iteration (RTI) scheme as introduced in Section 1.5. The numerical experiments presented in this section, have been performed using the `ACADO` code generation tool on an ordinary computer (Intel i7-3720QM 6MB cache, 2.60 GHz, 64-bit Ubuntu 12.04 and Clang 3.0).

4.3.1 NMPC on an Overhead Crane

We use a dynamic model very similar to the one in [76, 316] for the overhead crane system setup. Let us immediately write down the differential equations in the presented three-stage model format of Eq. (4.1). The linear input subsystem then consists of

$$\begin{aligned} \dot{x}_{\mathrm{T}} &= v_{\mathrm{T}}, \qquad \dot{v}_{\mathrm{T}} = a_{\mathrm{T}}, \qquad \dot{u}_{\mathrm{T}} = u_{\mathrm{TR}}, \\ \dot{x}_{\mathrm{L}} &= v_{\mathrm{L}}, \qquad \dot{v}_{\mathrm{L}} = a_{\mathrm{L}}, \qquad \dot{u}_{\mathrm{L}} = u_{\mathrm{LR}}, \end{aligned} \tag{4.11}$$

where $(x_{\mathrm{T}}, v_{\mathrm{T}})$ and $(x_{\mathrm{L}}, v_{\mathrm{L}})$ respectively denote the position and velocity of the trolley and the cable length for the overhead crane. Second order dynamics can be identified for the input-output relation of both the trolley (u_{T} to x_{T}) and the winching mechanism (u_{L} to x_{L}) where

$$a_{\mathrm{T}} = -\frac{1}{\tau_1} v_{\mathrm{T}} + \frac{A_1}{\tau_1} u_{\mathrm{T}}, \qquad a_{\mathrm{L}} = -\frac{1}{\tau_2} v_{\mathrm{L}} + \frac{A_2}{\tau_2} u_{\mathrm{L}}. \tag{4.12}$$

	Unstructured	With structure exploitation
Integration method	**220 µs**	**67 µs**
Condensing	6 µs	6 µs
QP solution (`qpOASES`)	16 µs	16 µs
Remaining operations	3 µs	3 µs
Total RTI step	245 µs	92 µs

Table 4.1: Average computation times: RTI based NMPC of an overhead crane.

Note that it is quite common in practical MPC implementations to include the change rates u_{TR}, u_{LR} of the system inputs in the dynamic model such that additional constraints on these variables can be included, to respect the limitations of the actuators [76]. The linear input system is then followed by these nonlinear differential equations

$$\begin{aligned} \dot{\theta} &= \omega, \\ \dot{\omega} &= -\frac{1}{x_{\mathrm{L}}}(g\sin(\theta) + a_{\mathrm{T}}\cos(\theta) + 2v_{\mathrm{L}}\omega), \end{aligned} \tag{4.13}$$

which describe the angle deflection θ and its angular velocity ω. Note that there is no linear output system in the original model formulation, for which the specific parameters can be found in [76, 316].

We now implement an NMPC algorithm, using the OCP formulation (1.6) based on the proposed model equations and a tracking type cost function such as in Eq. (1.25) or as described in [260] to achieve a point-to-point motion with the overhead crane system. Table 4.1 shows the computational speedup for one iteration of the Gauss-Newton based RTI scheme, implemented using the open-source `ACADO` code generation tool. With $N = 10$ control intervals over a horizon of $T = 1$ s and $N_{\mathrm{s}} = 4$ integration steps of the 4^{th} order Gauss collocation scheme within each interval, the table presents average computation times for the different components with and without the three-stage structure exploitation. For this relatively small and simple dynamic model, a speedup of factor 3 can already be observed in the total computation time spent in the numerical integration and sensitivity propagation.

We illustrated that a linear input subsystem appears naturally in the dynamic model for the overhead crane system [316]. Consider now the situation where one would like to additionally penalize the high frequency content in the states.

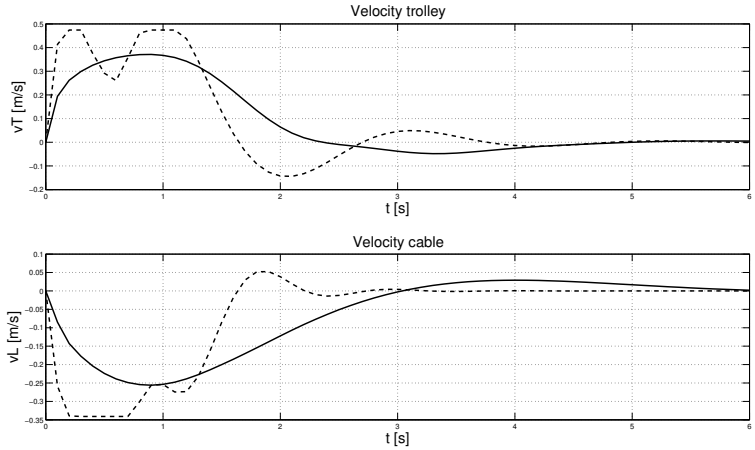


Figure 4.2: Closed-loop trajectories for the velocity of both trolley and cable, before (dashed) and after extra penalization of high frequencies (solid).

One way to do this is by implementing a high-pass filter such as in Eq. (4.5) for each of the differential states. This results in the following 6 equations being added to the linear input system

$$\begin{aligned} \dot{x}_{\mathrm{T}}^{\mathrm{HP}} &= \dot{x}_{\mathrm{T}} - \omega_{\mathrm{c}} x_{\mathrm{T}}^{\mathrm{HP}}, \qquad & \dot{x}_{\mathrm{L}}^{\mathrm{HP}} &= \dot{x}_{\mathrm{L}} - \omega_{\mathrm{c}} x_{\mathrm{L}}^{\mathrm{HP}}, \\ \dot{v}_{\mathrm{T}}^{\mathrm{HP}} &= \dot{v}_{\mathrm{T}} - \omega_{\mathrm{c}} v_{\mathrm{T}}^{\mathrm{HP}}, \qquad & \dot{v}_{\mathrm{L}}^{\mathrm{HP}} &= \dot{v}_{\mathrm{L}} - \omega_{\mathrm{c}} v_{\mathrm{L}}^{\mathrm{HP}}, \\ \dot{u}_{\mathrm{T}}^{\mathrm{HP}} &= \dot{u}_{\mathrm{T}} - \omega_{\mathrm{c}} u_{\mathrm{T}}^{\mathrm{HP}}, \qquad & \dot{u}_{\mathrm{L}}^{\mathrm{HP}} &= \dot{u}_{\mathrm{L}} - \omega_{\mathrm{c}} u_{\mathrm{L}}^{\mathrm{HP}}, \end{aligned} \tag{4.14}$$

but also in an extra linear output system consisting of

$$\dot{\theta}^{\mathrm{HP}} = \dot{\theta} - \omega_{\mathrm{c}} \theta^{\mathrm{HP}}, \qquad \dot{\omega}^{\mathrm{HP}} = \dot{\omega} - \omega_{\mathrm{c}}\, \omega^{\mathrm{HP}}. \tag{4.15}$$

The new differential states in these equations describe the high frequency content in the corresponding variables from the original system, leading to a complete set of 16 differential states of which 12 are in the linear input system, 2 nonlinear equations and 2 linear output states. Weighting these high frequency states in the NMPC formulation causes a certain smoothing of the closed-loop trajectories, as illustrated in Figure 4.2. Similarly, Table 4.2 now presents a speedup factor of about 6 for the numerical integration time due to the three-stage structure exploitation on this augmented dynamic model.

	Unstructured	With structure exploitation
Integration method	**1380 µs**	**238 µs**
Condensing	25 µs	25 µs
QP solution (`qpOASES`)	12 µs	12 µs
Remaining operations	13 µs	13 µs
Total RTI step	1430 µs	288 µs

Table 4.2: Average computation times for RTI based NMPC of an overhead crane, including the penalization of higher frequency state information.

4.3.2 NMPC on a Quadcopter System

In this second example, NMPC will be applied to a quadcopter (e.g., see [170]). We use a rather simple system setup, where it is assumed that low-level speed controllers ensure the tracking of the reference velocities of the propellers, which are then provided by the NMPC scheme. The differential equations can be briefly summarized as

$$\ddot{\omega}_k^{\text{ref}} = U_k, \quad k = 1, \dots, 4, \qquad \dot{\omega}_k = \tau^{-1} \left(\omega_k^{\text{ref}} - \omega_k \right), \tag{4.16a}$$

$$\dot{q} = \frac{1}{2} E^T \Omega, \qquad \dot{\Omega} = J^{-1} \left(T + \Omega \times J\Omega \right), \tag{4.16b}$$

$$\dot{v} = m^{-1} RF - g\mathbf{1}_z, \tag{4.16c}$$

$$F = \frac{1}{2} \sum_{k=1}^{4} \rho A C_{\text{L}} \omega_k^2 \mathbf{1}_z, \qquad T = \sum_{k=1}^{4} \frac{(-1)^k}{2} \rho A C_{\text{D}} \omega_k^2 \mathbf{1}_z, \tag{4.16d}$$

$$\dot{p} = v, \qquad \dot{I}_{\text{p}} = p - p_{\text{ref}}, \tag{4.16e}$$

where U_k for $k = 1, \dots, 4$ are the control inputs, commanding the 2^{nd} time derivative of the propeller reference velocities ω_k^{ref}. In addition, ω_k are the actual propeller velocities, $q \in \mathbb{R}^4$ is the quaternion vector used to represent the orientation, Ω is the main body angular velocity in the body frame, v is the linear velocity in the inertial frame, p is the position and $I_{\text{p}} \in \mathbb{R}^3$ the integral of the position error. Matrix $R = EG^T$ is the rotation matrix between the body

frame and the inertial frame, with

$$G = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \quad E = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix}.$$

Parameter τ is the time constant of the speed control loops, $J \in \mathbb{R}^{3\times3}$ is the inertia matrix of the quadcopter, and m its total mass. Coefficients C_{L} and C_{D} are respectively the lift and drag coefficients, A is the individual area of the propellers, ρ is the air density, and $\mathbf{1}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

It can be observed that the differential equations in (4.16a) form a 12 states linear input system, Eqs. (4.16b)-(4.16c) form a 10 states nonlinear dynamic system and Eq. (4.16e) corresponds to a 6 states linear output system. The input dynamics $\ddot{\omega}_k^{\mathrm{ref}} = U_k$ are used to implement a penalty on the high-frequency components in the controls, i.e., the Lagrange term

$$\Pi_{\mathrm{input}} = \sum_{k=1}^{4} W_1 U_k^2 + W_2 \left(\dot{\omega}_k^{\mathrm{ref}}\right)^2$$

is added to the objective, with positive weights W_1 and W_2. Moreover, the integral of the position error $\dot{I}_{\mathrm{p}} = p - p_{\mathrm{ref}}$ is introduced and penalized as a possible way to eliminate steady-state errors in the quadcopter position, resulting from constant disturbances or model errors.

The positive definite weighting matrices Q, R and P in the tracking type OCP cost function from Eq. (1.25) are chosen to achieve quick point-to-point motions of the quadcopter while taking into account its limitations. A possible closed-loop trajectory for the position and orientation of the quadcopter is presented in Figure 4.3, in the event of a motion from the point $(1, 1, 1)$ to the origin. It shows the position as well as the orientation of the quadcopter at multiple time instants, which are 0.4 s apart from each other. Table 4.3 additionally shows a computational speedup factor of about 12 for the numerical integration time. For these experiments, a horizon of length $T = 5$ s with $N = 25$ control intervals has been used and $N_{\mathrm{s}} = 2$ integration steps per interval of the 6^{th} order Gauss collocation scheme. For this specific case study, it can be observed that the structure exploitation seems to cause condensing to become the dominating computational cost in each RTI step.

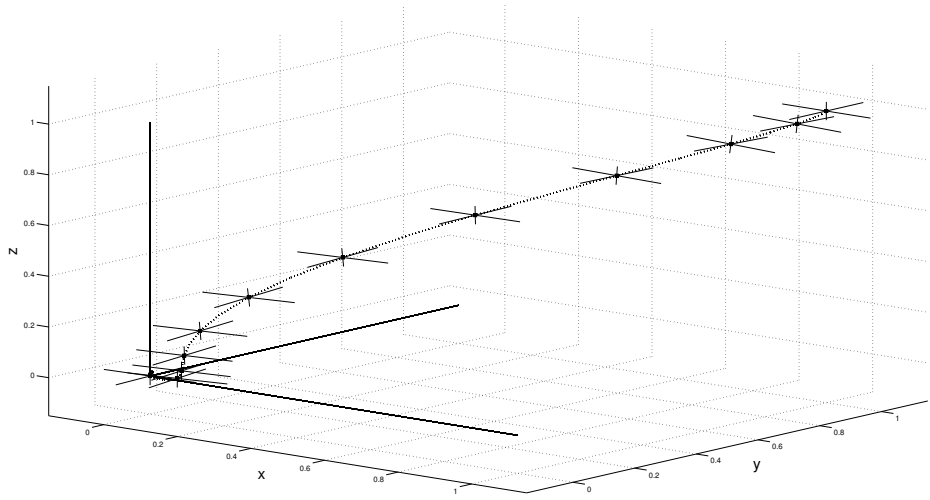


Figure 4.3: Illustration of a closed-loop trajectory of the position and orientation of the quadcopter in a point-to-point motion based on NMPC.

	Unstructured	With structure exploitation
Integration method	**35.8 ms**	**3.1 ms**
Condensing	2.6 ms	2.6 ms
QP solution (`qpOASES`)	0.1 ms	0.1 ms
Remaining operations	0.2 ms	0.2 ms
Total RTI step	38.7 ms	6.0 ms

Table 4.3: Average computation times for RTI based NMPC of a quadcopter.

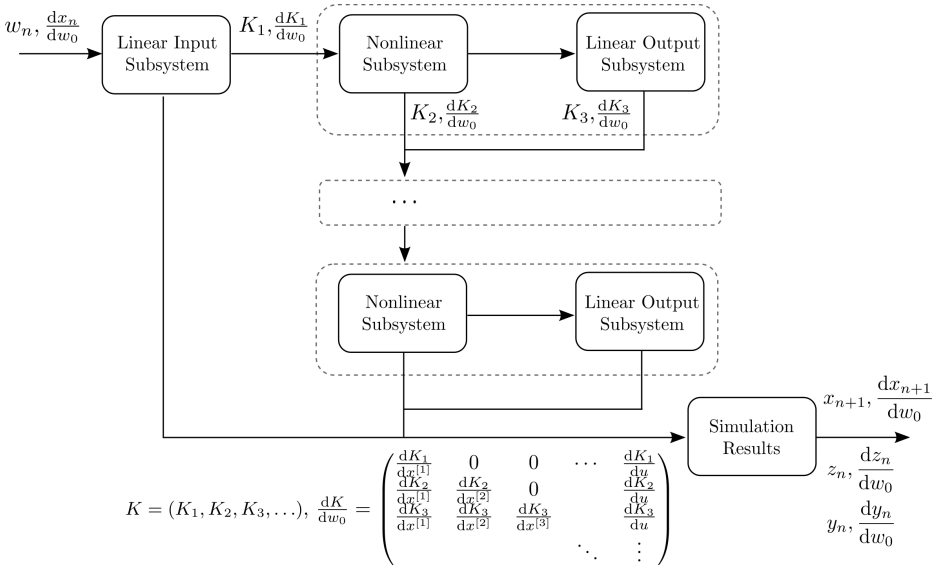


Figure 4.4: Schematic of a multi-stage dynamic system structure, illustrating the workflow in an extended structure exploiting collocation based integrator.

4.4 Conclusions and Outlook

This chapter proposed a new format for defining nonlinear dynamic models within direct optimal control and showed how this three-stage structure can be strongly exploited by IRK methods with a focus on collocation schemes. Numerical experiments of NMPC for two different case studies have shown that considerable speedups can be achieved with these structure exploiting integrators, which have been made available in the `ACADO Toolkit` [3]. It is interesting to note that these linear subsystems could be detected automatically inside any modeling environment with differentiation capabilities. Future work could include implementing such an automatic structure detection based on a smart restructuring of the equations. As argued in this chapter, the presented model formulation can however be more effective when taken into account directly at the modeling stage to reduce the relative amount of nonlinearly defined state variables. In addition, the proposed dynamic model could readily be extended to multiple stages of coupled nonlinear and linear output subsystems as illustrated also in Figure 4.4.

Chapter 5

Compression Algorithm for Distributed Multiple Shooting

The focus of the presented online algorithms for NMPC has been on centralized systems. However, many practical and fast dynamic systems can be described as a set of interconnected subsystems. Some examples include cars (decomposable into chassis and wheel dynamics), mobile robots with trailers (each trailer can be considered as a subsystem), quadcopters (decomposable into a rigid body and separate rotor dynamics), and many more. Distributed Multiple Shooting (DMS) [290] is a highly parallelizable discretization scheme for distributed systems. Exploiting the structure of the system, this approach extends the classical, centralized multiple shooting (CMS) method of separating the simulations in time by additionally distributing them in state space. This idea has been shown suitable for large-scale control problems of interconnected processes such as network systems or process plants [195, 289]. This chapter proposes a novel compression technique, which performs a tailored structure exploitation to efficiently treat each sparse QP subproblem in a centralized optimization framework. This allows one to adopt DMS into an RTI type online algorithm for the optimal control of fast dynamic systems, as illustrated also numerically for a sequential software implementation.

Note that this chapter is largely based on the article in [272].

Outline The chapter is organized as follows. Section 5.1 first introduces and motivates the Distributed Multiple Shooting (DMS) scheme. The novel compression technique is then detailed in Section 5.2, which allows the efficient

implementation of DMS based embedded optimization. The performance of the resulting algorithm is illustrated numerically in Section 5.3, using the nontrivial example of a chain of spring connected masses.

5.1 Distributed Multiple Shooting

In what follows, we refer to the classical variant as Centralized Multiple Shooting (CMS), as opposed to Distributed Multiple Shooting (DMS) presented in [290]. Let us consider a process that can be divided into M coupled subsystems. The following continuous time OCP can then be considered as a generalization of our standard parametric formulation in (1.6)

$$\min_{x(\cdot),u(\cdot),v(\cdot),y(\cdot)} \quad \sum_{j=1}^{M} \left(\int_0^T \ell^j(x^j(t), u^j(t))\,\mathrm{d}t + m^j(x^j(T)) \right) \tag{5.1a}$$

$$\text{s.t.} \qquad x^j(0) = \hat{x}_0^j, \tag{5.1b}$$

$$0 = f^j(\dot{x}^j(t), x^j(t), u^j(t), v^j(t)), \tag{5.1c}$$

$$y^j(t) = g^j(\dot{x}^j(t), x^j(t), u^j(t), v^j(t)), \tag{5.1d}$$

$$v^j(t) = \sum_{m=1}^{M} K_{jm} y^m(t), \tag{5.1e}$$

$$0 \geq h^j(x^j(t), u^j(t)), \tag{5.1f}$$

$$0 \geq r^j(x^j(T)), \quad \forall t \in [0,T], \quad j = 1, \ldots, M, \tag{5.1g}$$

where $y^j(t) \in \mathbb{R}^{n_{\mathrm{y},j}}$ denote the coupling outputs and $v^j(t) \in \mathbb{R}^{n_{\mathrm{v},j}}$ are the coupling inputs for each subsystem $j = 1, \ldots, M$. Function $f^j(\cdot)$ defines the nonlinear dynamics with the differential states $x^j(t) \in \mathbb{R}^{n_{\mathrm{x},j}}$ and control inputs $u^j(t) \in \mathbb{R}^{n_{\mathrm{u},j}}$ of subsystem $j = 1, \ldots, M$, while $g^j(\cdot)$ defines its output function. Each subsystem is described by an implicit ODE system in (5.1c) although this could readily be extended to an implicit index-1 DAE based on the techniques discussed in Chapter 2. Note that this parametric optimization problem depends on the current state estimate $\hat{x}_0^j \in \mathbb{R}^{n_{\mathrm{x},j}}$ via Eq. (5.1b) for each subsystem. For the sake of notational convenience, we restrict to the case where the objective (5.1a) and constraint functions in (5.1f)-(5.1g) are directly decoupled while they can generally depend on the coupling variables. The linear relation based on the coupling matrices K_{jm} in Eq. (5.1e) defines the specific

interaction among the subsystems, by connecting the inputs $v^j(t)$ to the outputs of all other subsystems $y^m(t)$ for $m = 1, \dots, M$. Note that $K_{jj} = 0$ can be assumed to hold for all $j = 1, \dots, M$, i.e., there is no self-coupling.

5.1.1 DMS and Coupling Parameterization

In direct optimal control, one first discretizes the OCP in (5.1) using N shooting intervals over the horizon $[0, T]$, which results in a discrete time problem formulation such as in Eq. (1.8). The resulting parametric NLP can be solved up to a local minimum, e.g., using an SQP-type algorithm. We further adopt the simplified formulation from Section 1.2.2, i.e., based on a piecewise constant control parameterization on an equidistant grid of N intervals. Unlike the case of the centralized multiple shooting method, here also the coupling variables $y^j(t)$ need to be discretized locally using a set of basis functions.

In the original DMS formulation [290], the coupling variables are parameterized using normalized and shifted Legendre polynomials. This strategy could be modeled using additional quadrature states [166] which can alternatively be represented by a linear output system as discussed in Chapter 4. Instead, we propose a computationally more efficient formulation that relies on a polynomial interpolation using the continuous output feature as introduced in Section 2.5. It results in the following parameterization

$$y^j(t) = \sum_{s=1}^{q} \ell_s(\tilde{t})\, y_{i,s}^j, \quad \text{with } \ell_s(\tilde{t}) = \prod_{r \neq s} \frac{\tilde{t} - c_r}{c_s - c_r}, \tag{5.2}$$

where c_s denotes the collocation points for $s = 1, \dots, q$ and $\ell_s(\tilde{t})$ are the Lagrange polynomials with $\tilde{t} = \frac{t - t_i}{t_{i+1} - t_i}$ over the shooting interval $t \in [t_i, t_{i+1}]$. Note that the values $y_{i,s}^j$ correspond to the numerical evaluation of the output variables $y^j(t)$ at the time points $t_i + c_s\, T_{\text{int}}$ for $s = 1, \dots, q$. By increasing the order q of the collocation polynomial, an arbitrarily high accuracy can be obtained for the coupling approximation. Based on this representation of the output variables, the system inputs are defined directly by the linear coupling relation in Eq. (5.1e). Therefore, the dynamics can be simulated independently for each subsystem $j = 1, \dots, M$ and each shooting interval $i = 0, \dots, N - 1$, and this, e.g., using a collocation scheme.

In what follows, let us collectively refer to the coupling variables for each subsystem $j = 1, \dots, M$ over a certain shooting interval $i = 0, \dots, N - 1$ by using the notation $y_i^j := [y_{i,1}^{j\top}, \dots, y_{i,q}^{j\top}]^\top$ and $v_i^j := [v_{i,1}^{j\top}, \dots, v_{i,q}^{j\top}]^\top$. Unlike the centralized NLP in (1.8), we can now propose the following DMS based discrete

time OCP formulation

$$\min_{X,U,V,Y} \quad \sum_{j=1}^{M} \sum_{i=0}^{N-1} l^j(x_i^j, u_i^j) + \sum_{j=1}^{M} m^j(x_N^j) \tag{5.3a}$$

$$\text{s.t.} \qquad 0 = x_0^j - \hat{x}_0^j, \tag{5.3b}$$

$$0 = x_{i+1}^j - \phi^j(x_i^j, u_i^j, v_i^j), \tag{5.3c}$$

$$0 = y_i^j - \psi^j(x_i^j, u_i^j, v_i^j), \tag{5.3d}$$

$$0 = v_i^j - \sum_{m=1}^{M} K_{jm}\, y_i^m, \tag{5.3e}$$

$$0 \geq h^j(x_i^j, u_i^j), \tag{5.3f}$$

$$0 \geq r^j(x_N^j), \qquad i = 0, \ldots, N-1, \quad j = 1, \ldots, M, \tag{5.3g}$$

using the differential state $X = [x_0^{1\top}, \ldots, x_0^{M\top}, \ldots, x_N^{1\top}, \ldots, x_N^{M\top}]^\top$ and control trajectory $U = [u_0^{1\top}, \ldots, u_0^{M\top}, \ldots, u_{N-1}^{1\top}, \ldots, u_{N-1}^{M\top}]^\top$, and similarly the discretized trajectories for coupling inputs V and outputs Y. Note that the discrete time dynamics for each subsystem in (5.3c) are based on a separate numerical integration scheme, which needs to evaluate the continuous time representation of the coupling variables in Eq. (5.2). Further in this chapter, we propose an efficient algorithmic technique to solve the resulting large but structured nonlinear optimization problem in (5.3).

Remark 5.1 *Note that if one step of a collocation method is used to simulate each subsystem in Eq.* (5.3c) *and the same collocation order q is used for the parameterization of the coupling variables in* (5.2)*, then the resulting distributed simulation scheme coincides with its centralized counterpart.*

5.1.2 Discussion and Motivation: DMS versus CMS

Just as CMS lifts the temporal continuity of the system equations in (1.8c), such that the numerical simulation can be carried out independently on each shooting interval, DMS additionally lifts the spatial continuity by keeping the subsystems independent for simulation. The result is a DMS scheme based on the NLP in Eq. (5.3) with the following desirable properties:

- It is highly parallelizable since the integration and sensitivity propagation on each interval and for each subsystem can be performed independently.

- Less sensitivity information needs to be computed in case of many subsystems with only relatively few coupling variables.
- When using an implicit integrator, the computational complexity can reduce substantially as shown further in Table 5.1.
- Different integration schemes can be used to, e.g., appropriately deal with stiffness within specific subsystems where necessary.
- Any form of lifting provides more initialization flexibility and can offer advantages in terms of local convergence and region of attraction [8].
- Tailored optimization algorithms can be used to exploit the specific sparsity structure resulting from DMS, as discussed in the next section.

Remark 5.2 *Since DMS is merely a different way of discretizing the same OCP, it can provide the exact same properties for the resulting NMPC algorithm as long as this discretization is carried out accurately enough. Note that the same condition holds for the more widely used CMS approach.*

Remark 5.3 *The DMS and CMS based optimal control solutions coincide for a sufficiently high simulation and coupling accuracy, even though the intermediate iterations of the optimization algorithm are generally different.*

5.2 The Compression Algorithm

When dealing with the DMS based OCP formulation from Eq. (5.3) in an SQP type framework (see Section 1.3.3), one needs to solve large but structured quadratic subproblems that include both the states and controls as well as all the coupling variables. This section proposes a compression technique which numerically eliminates the coupling variables in order to obtain a standard optimal control structured QP as in Eq. (1.22). In combination with a tailored convex solver such as `qpOASES` [109], `FORCES` [97], `qpDUNES` [117] or `HPMPC` [123], this approach allows one to efficiently exploit the coupling structure. The idea here is to pursue a minimal computational delay between obtaining the new state estimate and applying the next control input to the system, because the compression algorithm can be made part of the preparation phase of the RTI scheme as discussed earlier in Section 1.5.5. A possible alternative would be to develop tailored QP algorithms to directly deal with this specific coupling structure, for example, as discussed in [196].

5.2.1 The Coupled Subproblem Structure

Let us look at a way of numerically eliminating the coupling variables from the QP subproblem without changing the SQP iterations, which will be referred to as condensing or *compression*. The linearization of the equality constraints in (5.3c)–(5.3e) at the current values $(\bar{x}_i^j, \bar{u}_i^j, \bar{v}_i^j, \bar{y}_i^j)$ for $i = 0, \ldots, N-1$ and $j = 1, \ldots, M$, has the following form

$$\Delta x_{i+1}^j = c_i^j + C_i^j \begin{bmatrix} \Delta x_i^j \\ \Delta u_i^j \end{bmatrix} + E_i^j \Delta v_i^j, \tag{5.4a}$$

$$\Delta y_i^j = d_i^j + D_i^j \begin{bmatrix} \Delta x_i^j \\ \Delta u_i^j \end{bmatrix} + F_i^j \Delta v_i^j, \tag{5.4b}$$

$$\Delta v_i^j = \sum_{m=1}^{M} K_{jm} \Delta y_i^m, \tag{5.4c}$$

where $\Delta x_i^j = x_i^j - \bar{x}_i^j$, $\Delta u_i^j = u_i^j - \bar{u}_i^j$ and similarly for Δv_i^j and Δy_i^j. For notational convenience, the state and control variables can be collectively referred to as $w_i^j := (x_i^j, u_i^j)$ or $\Delta w_i^j := (\Delta x_i^j, \Delta u_i^j)$. In addition, the values $c_i^j = \phi^j(\bar{x}_i^j, \bar{u}_i^j, \bar{v}_i^j) - \bar{x}_{i+1}^j$, $d_i^j = \psi^j(\bar{x}_i^j, \bar{u}_i^j, \bar{v}_i^j) - \bar{y}_i^j$ and the Jacobian matrices $C_i^j = \frac{\partial \phi^j}{\partial w_i^j}(\bar{w}_i^j, \bar{v}_i^j)$, $D_i^j = \frac{\partial \psi^j}{\partial w_i^j}(\bar{w}_i^j, \bar{v}_i^j)$ and $E_i^j = \frac{\partial \phi^j}{\partial v_i^j}(\bar{w}_i^j, \bar{v}_i^j)$, $F_i^j = \frac{\partial \psi^j}{\partial v_i^j}(\bar{w}_i^j, \bar{v}_i^j)$ are defined. By eliminating the variables Δv_i^j based on the coupling relation in (5.4c), we obtain the following linear system

$$G_i \begin{bmatrix} \Delta y_i^1 \\ \vdots \\ \Delta y_i^M \end{bmatrix} = \begin{bmatrix} d_i^1 \\ \vdots \\ d_i^M \end{bmatrix} + \begin{bmatrix} D_i^1 & & \\ & \ddots & \\ & & D_i^M \end{bmatrix} \begin{bmatrix} \Delta w_i^1 \\ \vdots \\ \Delta w_i^M \end{bmatrix}, \tag{5.5}$$

where the coupling matrix G_i is defined as

$$G_i = \begin{bmatrix} \mathbb{1} & -F_i^1 K_{12} & \ldots & -F_i^1 K_{1M} \\ -F_i^2 K_{21} & \mathbb{1} & & \\ \vdots & & \ddots & \vdots \\ -F_i^M K_{M1} & \ldots & & \mathbb{1} \end{bmatrix}. \tag{5.6}$$

This means that one can numerically eliminate all coupling variables at the cost of factorizing the structured matrix G_i, such that one can rewrite the linearized equation for Δy_i^j from (5.4b) as follows

$$\Delta y_i^j = \tilde{d}_i^j + \sum_{m=1}^{M} \tilde{D}_i^{j,m} \Delta w_i^m, \tag{5.7}$$

where the vectors $\tilde{d}_i^j$ and matrices $\tilde{D}_i^{j,m}$ are the result of a compression routine as discussed in the next Subsection. One can substitute the coupling variables in (5.4a), using these expressions and the coupling graph as defined in (5.4c). The result is a compressed convex subproblem that is of the same form as the standard optimal control structured QP from Eq. (1.22).

Remark 5.4 *Regardless of how each DMS structured subproblem is solved, the same SQP iterations are performed in the full variable space* $(x_i^j, u_i^j, v_i^j, y_i^j)$ *where* i *and* j *are respectively the shooting and subsystem index. Given the step* $(\Delta x_i^j, \Delta u_i^j)$ *from the QP solution, the corresponding update for the coupling variables* $(\Delta y_i^j, \Delta v_i^j)$ *can be computed from Eqs.* (5.7) *and* (5.4c)*. This additional procedure is better known as the expansion step [8].*

Remark 5.5 *Note that the compression and expansion procedure can be carried out independently and therefore in parallel for each shooting interval.*

5.2.2 Structure Exploiting Compression

The sensitivity information is propagated for each subsystem separately resulting in the block diagonal right-hand side of Eq. (5.5). More importantly, the coupling matrix G_i itself is often rather sparse as each subsystem is typically coupled with just a few other subsystems. This block sparsity structure can inspire the use of, e.g., a block LU factorization as discussed in [78]. A tailored implementation could naturally take specific block structures into account. Note that a block LU factorization can be shown to be numerically stable when the matrix G is block diagonally dominant by columns, i.e.,

$$\|G_{jj}^{-1}\|^{-1} \geq \sum_{i=1, i\neq j}^{M} \|G_{ij}\|, \quad \forall j = 1, \ldots, M, \tag{5.8}$$

where G_{ij} are defined as the blocks in (5.6). Note that the diagonal blocks are identity matrices, while the off-diagonal blocks define the coupling with other subsystems. That motivates the idea that this assumption (5.8) typically holds for the presented DMS case. Otherwise, the coupling matrix is often well-conditioned which bounds the level of instability as discussed in [78].

Let us briefly focus on block tridiagonal matrices, as the block LU factorization is often used for such matrices. They arise frequently in the discretization of partial differential equations (PDE) [307] and this particular structure will also be observed in our case study from Section 5.3. The block tridiagonal coupling

Algorithm 4 Block Tridiagonal Compression

Input: coupling matrix G_i from (5.9) and values d_i^j, D_i^j from (5.5).
Output: compression results $\tilde{d}_i^j$, $\tilde{D}_i^{j,1:M}$.

`Forward procedure`

$\tilde{d}_i^1 \leftarrow d_i^1$, $\tilde{D}_i^{1,1} \leftarrow D_i^1$ and $\tilde{\bar{F}}_i^1 \leftarrow \bar{F}_i^1$.
for $j = 2 \rightarrow M$ **do**
  compute factorization of $H_j = (\mathbb{1} - \underline{F}_i^j \tilde{\bar{F}}_i^{j-1})$.
  $\tilde{d}_i^j \leftarrow H_j^{-1} d_i^j$ and $\tilde{D}_i^{j,j} \leftarrow H_j^{-1} D_i^j$.
  $\tilde{\underline{F}}_i^j \leftarrow H_j^{-1} \underline{F}_i^j$ and $\tilde{\bar{F}}_i^j \leftarrow H_j^{-1} \bar{F}_i^j$.
  $\tilde{d}_i^j \leftarrow \tilde{d}_i^j + \tilde{\underline{F}}_i^j \tilde{d}_i^{j-1}$ and $\tilde{D}_i^{j,1:j-1} \leftarrow \tilde{\underline{F}}_i^j \tilde{D}_i^{j-1,1:j-1}$.
end for

`Back substitution`

for $j = M - 1 \rightarrow 1$ **do**
  $\tilde{d}_i^j \leftarrow \tilde{d}_i^j + \tilde{\bar{F}}_i^j \tilde{d}_i^{j+1}$ and $\tilde{D}_i^{j,1:M} \leftarrow \begin{bmatrix} \tilde{D}_i^{j,1:j} & \mathbb{0} \end{bmatrix} + \tilde{\bar{F}}_i^j \tilde{D}_i^{j+1,1:M}$.
end for

matrix looks as follows

$$G_i = \begin{bmatrix} \mathbb{1} & -\bar{F}_i^1 & & & \mathbb{0} \\ -\underline{F}_i^2 & \mathbb{1} & -\bar{F}_i^2 & & \\ & & \ddots & & -\bar{F}_i^{M-1} \\ \\ \mathbb{0} & & & -\underline{F}_i^M & \mathbb{1} \end{bmatrix}, \tag{5.9}$$

where $\bar{F}_i^j = F_i^j K_{j,j+1}$ and $\underline{F}_i^j = F_i^j K_{j,j-1}$. This corresponds to a DMS type OCP in which the different subsystems are coupled in the form of a linear graph. Algorithm 4 presents the compression method based on a block tridiagonal LU factorization as presented in [137]. Note that a compact notation is used where $\tilde{D}_i^{j,1:M}$ denotes the block matrices $\tilde{D}_i^{j,k}$ for $k = 1, \ldots, M$. As an alternative to using tailored block factorizations, also a general-purpose sparse linear solver could be used to implement the compression technique.

5.2.3 Computational Complexity

For the sake of simplicity, we consider M subsystems with an equal amount of n_x states and n_y coupling variables, which denotes both the number of coupling in- and outputs. It is additionally assumed that $N_\mathrm{s} = 1$ integration step of the q-stage collocation scheme is performed and also the order of the coupling

Table 5.1: Computational complexity analysis based on a q-stage collocation scheme: CMS versus DMS with compression ($n_{\mathrm{w}} = n_{\mathrm{x}} + n_{\mathrm{u}}$).

	CMS	DMS
Simulation and sensitivities	$(q\,n_{\mathrm{x}}M)^3 + (q\,n_{\mathrm{x}}M)^2 M n_{\mathrm{w}}$	$(q\,n_{\mathrm{x}})^3 \times M + (q\,n_{\mathrm{x}})^2(n_{\mathrm{w}} + q\,n_{\mathrm{y}}) \times M$
Dense (5.6) compression	-	$(q\,n_{\mathrm{y}}M)^3 + (q\,n_{\mathrm{y}}M)^2\,M n_{\mathrm{w}}$
Tridiagonal (5.9) compression	-	$(q\,n_{\mathrm{y}})^3 \times M + (q\,n_{\mathrm{y}})^2(M n_{\mathrm{w}} + q\,n_{\mathrm{y}}) \times M$

parameterization is equal to q. Depending on the structure of the coupling matrix, the tridiagonal method from Algorithm 4 can be used.

Table 5.1 then presents an asymptotic comparison in computational complexity of applying the q-stage collocation method in both the CMS and DMS framework. The latter allows one to simulate and propagate sensitivities for the different subsystems separately but it requires the use of compression. The table focuses on matrix factorizations and the corresponding back substitutions, since they are typically the dominating cost factor. The computational complexity analysis is based on the exact Newton implementation in Table 2.1, in combination with a direct sensitivity propagation which does not depend on the number of iterations. We further also omit constant factors and instead look at the asymptotic complexity behaviour. It can be seen from this table that the computational cost could become substantially less in the distributed case depending on the polynomial order q, the number of subsystems M and on the ratio of coupling variables $n_{\mathrm{y}} < n_{\mathrm{x}}$. It becomes however difficult to draw more general conclusions, since the comparison depends highly on the specific coupling structure between the different subsystems.

5.3 NMPC Application: Chain of Masses

The complexity analysis presented in Table 5.1 states that the advantage of using DMS instead of CMS is greater for larger systems composed of many small and weakly coupled subsystems. This section illustrates the performance of a DMS based real-time NMPC implementation on returning a chain of spring connected masses to its steady state, e.g., similar to the case study from [325]. Figure 5.1 illustrates the considered setup, which consists of 8 masses forming a

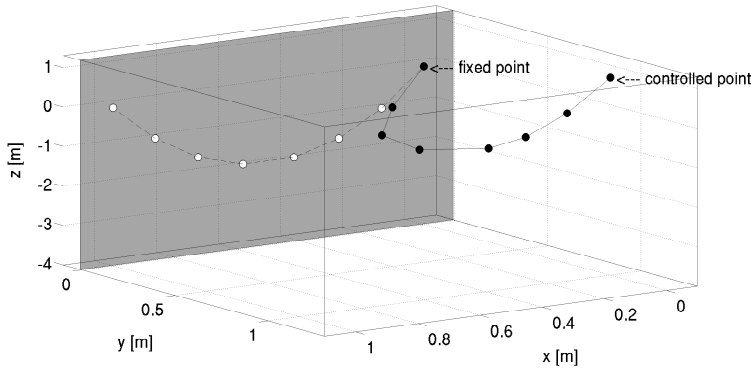


Figure 5.1: Benchmark case study example for optimal control: illustration of a chain of $n_{\mathrm{m}} = 8$ masses connected by springs.

chain with a fixed and a directly controlled end. The goal is to move the chain from a given initial configuration to its equilibrium state without touching a closely positioned wall (depicted in gray). The system exhibits fast nonlinear dynamics due to the spring forces and therefore a horizon length of $T = 4$ s and a sampling time of $T_s = 0.2$ s is chosen.

All numerical simulations are performed using the `ACADO` code generation tool on a standard computer equipped with Intel i7-3720QM processor. Although for simplicity, only the simulation results of a sequential implementation will be shown, let us underline that the presented DMS based algorithm can be straightforwardly parallelized.

5.3.1 System Dynamics and Coupling Structure

We consider a chain of $n_{\mathrm{m}} = 8$ elements with mass m, connected by springs with rest length L and spring constant D. One end of the chain is fixed at point $x_0 = [0, 0, 0]^\top$, as illustrated in Figure 5.1. Let us denote the positions $x_i(t) \in \mathbb{R}^3$ and the velocities $v_i(t) \in \mathbb{R}^3$ for the free masses $i = 1, \ldots, 7$. The equations for the controlled end of the chain read

$$\dot{x}_7(t) = v_7(t), \qquad \dot{v}_7(t) = u(t),$$

where $u(t) \in \mathbb{R}^3$ denotes the control inputs of the system. For each of the remaining masses $i = 1, \ldots, 6$, the following dynamics hold

$$\dot{x}_i(t) = v_i(t), \quad \dot{v}_i(t) = \frac{1}{m}\left(F_{i,i+1} - F_{i-1,i}\right) + g,$$

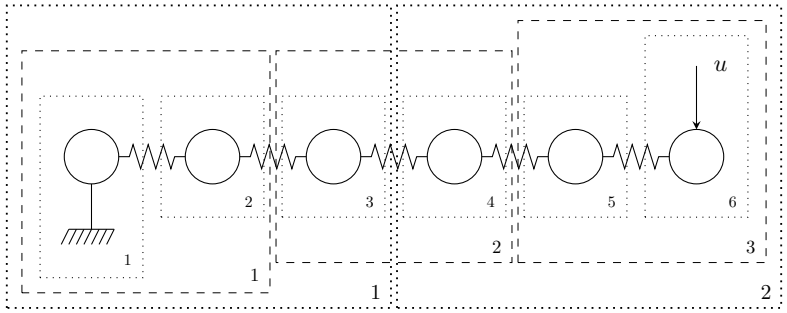


Figure 5.2: Chain mass optimal control example: illustration of the different partitioning options to identify the coupled subsystems.

where $g \in \mathbb{R}^3$ is the gravitational acceleration and $F_{i,i+1} \in \mathbb{R}^3$ denotes the force acting on mass i due to the spring between mass i and mass $i+1$. For more details on the nonlinear expressions for this dynamic model and the corresponding parameter values, we refer to [325].

A natural way of partitioning this system into subsystems consists in defining each mass as a separate subsystem. The linear dynamics of the controlled end of the chain can be efficiently handled within a collocation based integration scheme as described in Chapter 4. Therefore, these linear dynamics are further considered as a part of the adjacent subsystem, resulting in 6 subsystems in total in that case. There are however different partitioning choices possible as illustrated also in Figure 5.2. Instead of defining 6 subsystems of which each describe one free mass, one can alternatively define 2 or 3 subsystems describing respectively 3 or 2 masses. The positions of the neighbouring masses can be identified as the coupling variables which are needed to compute the corresponding spring forces in the dynamics. The resulting coupling structure in the form of a linear graph allows for the use of the block tridiagonal compression method from Algorithm 4.

5.3.2 Numerical Simulation Results

The optimal control objective minimizes the deviation of the mass positions and velocities from the equilibrium state and this in the form of a least squares type cost function. We therefore use the Gauss-Newton based RTI scheme as introduced in Section 1.5 for Nonlinear MPC, and this based on the open-source implementation in the `ACADO` code generation tool. Figure 5.3 shows the results of a closed-loop NMPC simulation for the chain of masses, starting from a

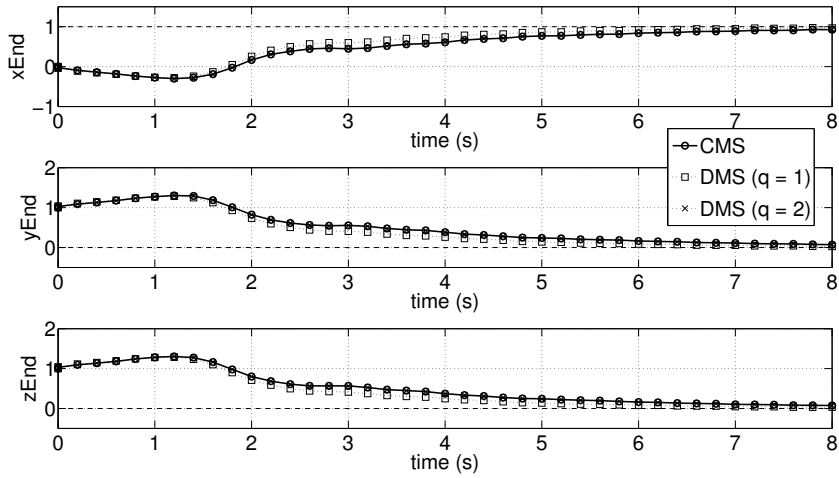


Figure 5.3: Comparison of the NMPC closed-loop trajectories for a CMS and a DMS based direct optimal control implementation.

given initial state. It shows the closed-loop trajectories for the position of the controlled end, using an `ACADO` generated RTI scheme both in the case of a CMS and a DMS based discretization. The OCP formulation includes a constraint for each of the masses not to hit the wall as illustrated in Figure 5.1 and control bounds $u_i \in [-10, 10],\ i = 1, 2, 3$. Note that, for a collocation coupling order $q \geq 2$, there is little noticeable difference between the CMS and the DMS trajectories. This is confirmed by the closed-loop cost values given in Table 5.2, which show little to no performance loss.

Table 5.2 shows the average computation time for the different components in a serial implementation of the RTI scheme using a CMS or a DMS based discretization. A fourth order Gauss collocation method with continuous output has been used here to perform the numerical simulation and implement the coupling of the $M = 6$ subsystems. The corresponding coupling variables in the large-scale QP subproblem can be eliminated using the proposed compression technique from Algorithm 4. The resulting standard optimal control structured QP is then solved using condensing in combination with `qpOASES`. Note that the feedback delay of the RTI scheme then corresponds to only the time spent in `qpOASES` to solve the small condensed QP, while the numerical simulation and sensitivity propagation, the compression and condensing procedure are all part of the preparation phase. In the case of DMS based optimal control with a

Table 5.2: Average computation times for CMS and DMS based RTI schemes on the optimal control example of the chain of masses.

	CMS	DMS		
		$q = 1$	$q = 2$	$q = 3$
integration	35.66 ms	1.82 ms	2.57 ms	3.61 ms
compression	-	0.22 ms	**0.56 ms**	1.07 ms
condensing	3.16 ms	3.16 ms	3.16 ms	3.16 ms
`qpOASES`	0.12 ms	0.12 ms	0.12 ms	0.12 ms
RTI step	38.94 ms	5.32 ms	**6.41 ms**	7.96 ms
closed-loop cost	717 [-]	724 [-]	717 [-]	717 [-]

Table 5.3: Average computation times for the compression procedure in the DMS implementation: Algorithm 4 versus a dense linear solver.

	$q = 1$	$q = 2$	$q = 3$
Algorithm 4	0.22 ms	0.56 ms	1.07 ms
dense solver	0.42 ms	1.83 ms	4.97 ms

coupling parameterization $q = 2$, integration together with compression takes 3.13 ms in total which is about 11 times faster than the case for the classical CMS approach. Regarding the total time for one RTI step, a corresponding speedup factor of about 6 can be observed.

Table 5.2 already illustrated the numerical performance of our proposed DMS based RTI scheme and its scaling with the order of the coupling approximation. Even when using an off-the-shelf dense solver instead of the block tridiagonal compression method, Table 5.3 shows that DMS combined with compression can provide considerable speedups in the total computation time. In the case of DMS based RTI with $q = 2$, integration together with dense compression takes 4.4 ms in total which results in a speedup factor 8. An interesting comparison in computation times between the division into $M = 6$ subsystems and the alternative options from Figure 5.2 can be found in Table 5.4. It illustrates the trade-off between the amount of small subsystems and the computational cost of the compression technique, even though the DMS-RTI scheme using $M = 6$ performs the best for our case study here.

Table 5.4: Average timing results for DMS based RTI using different partitions for the chain of masses in coupled subsystems ($q = 2$).

	$M = 1$	$M = 2$	$M = 3$	$M = 6$
integration	35.66 ms	9.08 ms	4.55 ms	2.57 ms
compression	-	0.22 ms	0.39 ms	0.56 ms
RTI step	38.94 ms	12.58 ms	8.22 ms	6.41 ms

5.4 Conclusions and Outlook

This chapter presented an efficient DMS based RTI algorithm to perform real-time NMPC for decomposable systems, including an approach to model the coupling between subsystems and a tailored compression technique to reduce the computational burden for the embedded convex solver. Already in a serial implementation, which serves as a proof of concept, the proposed approach is shown to provide impressive speedups over the conventional scheme for certain dynamic systems. This technique could be combined with the three- or even multi-stage model structure from Chapter 4 within each subsystem, to result in a rather flexible algorithmic framework with tailored structure exploitation. This work can also be considered as an important step towards a real-time fully distributed online NMPC algorithm. The promising combination of the DMS scheme with the highly parallelizable "Augmented Lagrangian based Alternating Direction Inexact Newton" (ALADIN) method [178] for direct optimal control [194], is part of ongoing research.

Chapter 6

Lifted Newton-Type Collocation Integrators

Popular approaches to tackle the continuous time OCP in Eq. (1.6) are multiple shooting [51] and direct collocation [36], which both treat the simulation and optimization problem simultaneously. A Newton-type algorithm is then able to find a locally optimal solution for the resulting NLP by solving the KKT optimality conditions [232]. When an implicit integration scheme is used for either stiff or implicitly defined dynamics within direct multiple shooting, one needs to implement an iterative Newton scheme for the integrator which is used within the Newton-type optimization algorithm. This chapter proposes a novel approach for embedding these implicit integrators within a Newton-type optimization framework, based on an extension of the lifted Newton method [8] for implicitly defined variables. An important advantage of the lifted collocation approach over direct collocation is that one can instead solve subproblems having the structure and dimensions of the multiple shooting method, for which efficient embedded solvers exist, based on dense linear algebra routines such as `qpOASES` [109], `FORCES` [96], `qpDUNES` [117] and `HPMPC` [123].

Note that this chapter is largely based on the article in [265], which itself is built upon earlier publications as part of multiple conference proceedings. The lifted collocation integrator was first presented in [263], followed by an extension to exact Hessian based optimization by using a symmetric forward-backward propagation technique as discussed in [267]. In addition, it was first proposed in [262] that the lifted collocation approach can be combined with the use of efficient inexact Newton-type methods.

Outline The chapter is organized as follows. Section 6.1 briefly discusses Newton-type optimization for both multiple shooting and direct collocation. The exact lifted collocation integrator for direct multiple shooting is presented in Section 6.2, including a detailed discussion of its advantages and disadvantages. Section 6.3 proposes a Newton-type optimization approach based on inexact lifted collocation and an adjoint derivative propagation. Advanced inexact lifted collocation methods based on an inexact Newton scheme with iterated sensitivities (INIS) are discussed in Section 6.4. Section 6.5 briefly describes the open-source software implementation of the proposed algorithms in the `ACADO` code generation tool, followed by a numerical case study in Section 6.6.

6.1 Simultaneous Direct Optimal Control

The discussions further can be easily extended to a general OCP formulation such as in Eq. (1.6), including an index 1 DAE and a terminal cost or terminal constraint. However, for the sake of simplicity, we omit these cases in the following, and even dismiss the path constraints (1.6d) without loss of generality for the presentation of the lifted collocation integrators which deal mostly with the system dynamics. For the problem discretization, we consider here an equidistant grid over the control horizon consisting of the collection of time points $t_{i+1} - t_i = \frac{T}{N} =: T_s$ for $i = 0, \dots, N-1$. Additionally, we consider a piecewise constant control parameterization $u(\tau) = u_i$ for $\tau \in [t_i, t_{i+1})$.

6.1.1 Collocation based Numerical Simulation

This chapter considers the use of a collocation scheme as introduced in Section 2.2.2, in order to compute a numerical approximation of the terminal state $x(t_{i+1})$ of the following initial value problem

$$0 = f(\dot{x}(\tau), x(\tau), u_i), \quad \tau \in [t_i, t_{i+1}], \quad x(t_i) = x_i, \tag{6.1}$$

where the dynamic system described by $f(\cdot)$ could be stiff. Based on the general IRK formulation in Eq. (2.28), let us introduce the system of collocation equations for a fixed number N_s of integration steps such that $T_\mathrm{int} := \frac{T_s}{N_\mathrm{s}}$. We use the notation as illustrated in Figure 6.1 for one shooting interval $i = 0, \dots, N-1$. To obtain the variables describing the collocation polynomials over the consecutive integration steps, one needs to solve the following system

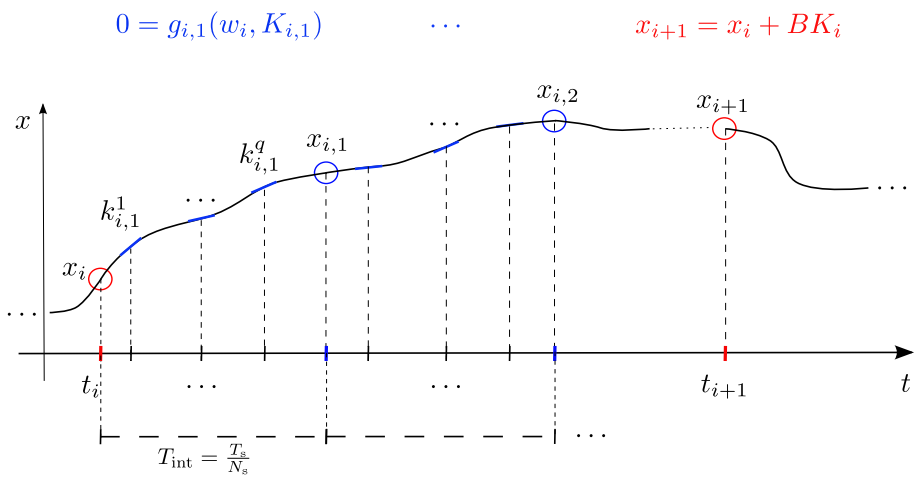


Figure 6.1: Illustration of N_{s} fixed integration steps of a collocation scheme over one shooting interval $[t_i, t_{i+1}]$, including the corresponding equations.

of collocation equations

$$G(w_i, K_i) = \begin{bmatrix} g_{i,1}(w_i, K_{i,1}) \\ \vdots \\ g_{i,N_{\text{s}}}(w_i, K_{i,1}, \dots, K_{i,N_{\text{s}}}) \end{bmatrix} = 0, \tag{6.2}$$

$$\text{where} \quad g_{i,j}(\cdot) = \begin{bmatrix} f(k^1_{i,j}, x_{i,j-1} + T_{\text{int}} \sum_{s=1}^{q} a_{1,s} k^s_{i,j}, u_i) \\ \vdots \\ f(k^q_{i,j}, x_{i,j-1} + T_{\text{int}} \sum_{s=1}^{q} a_{q,s} k^s_{i,j}, u_i) \end{bmatrix},$$

where $w_i := (x_i, u_i)$, q denotes the number of collocation nodes and the matrix $[A]_{ij} := a_{i,j}$ contains the coefficients of the method [158]. The collocation variables $k^s_{i,j} \in \mathbb{R}^{n_{\text{x}}}$ are collectively denoted by $K_i := (K_{i,1}, \dots, K_{i,N_{\text{s}}}) \in \mathbb{R}^{n_{\text{K}}}$ with $K_{i,j} := (k^1_{i,j}, \dots, k^q_{i,j})$ for $i = 0, \dots, N-1$ and $j = 1, \dots, N_{\text{s}}$. The intermediate state values $x_{i,j}$ are defined by the collocation variables and by the weights b_s of the q-stage method

$$x_{i,j} = x_{i,j-1} + T_{\text{int}} \sum_{s=1}^{q} b_s k^s_{i,j}, \quad j = 1, \dots, N_{\text{s}}, \tag{6.3}$$

where $x_{i,0} = x_i$. The simulation result can then be obtained as $x_{i,N_{\text{s}}} = x_i + B\,K_i$ in which B is a constant matrix that depends on the fixed step size T_{int} and the variables K_i satisfy the collocation equations $G(w_i, K_i) = 0$.

6.1.2 Simultaneous Direct Optimal Control

As a special case of Eq. (1.8), the direct *multiple shooting* discretization [51] of an equality constrained OCP results in the NLP

$$\min_{X,\,U} \quad \sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) \tag{6.4a}$$

$$\text{s.t.} \quad 0 \;=\; x_0 - \hat{x}_0, \tag{6.4b}$$

$$0 \;=\; \phi(x_i, u_i) - x_{i+1}, \quad i = 0, \ldots, N-1, \tag{6.4c}$$

with state $X = [x_0^\top, \ldots, x_N^\top]^\top$ and control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$. In what follows, all the optimization variables for this NLP (6.4) can also be referred to as the concatenated vector $W = [x_0^\top, u_0^\top, \ldots, x_N^\top]^\top \in \mathbb{R}^{n_\mathrm{W}}$ where $n_\mathrm{W} = n_\mathrm{x} + N(n_\mathrm{x} + n_\mathrm{u})$. In the case of a fixed step collocation method, the function $\phi(\cdot)$ can be defined as

$$\phi(x_i, u_i) = x_i + B\, K_i(x_i, u_i), \tag{6.5}$$

where the collocation variables are obtained implicitly by solving the system of equations in (6.2) which depends on the state value x_i and control input u_i. The Lagrangian of the NLP in (6.4) is given by

$$\begin{aligned}\mathcal{L}(W, \Lambda) &= \sum_{i=0}^{N-1} \mathcal{L}_i(w_i, \lambda_i) + m(x_N) \\ &= \sum_{i=0}^{N-1} l(w_i) + m(x_N) + \lambda_{-1}^\top (x_0 - \hat{x}_0) + \sum_{i=0}^{N-1} \lambda_i^\top \left(\phi(w_i) - x_{i+1}\right),\end{aligned} \tag{6.6}$$

where λ_i for $i = 0, \ldots, N-1$ denote the multipliers corresponding to the continuity constraints in (6.4c) and λ_{-1} denotes the multiplier of the initial value condition (6.4b).

A popular alternative to multiple shooting is *direct collocation* [36], which is an example of a direct transcription method as introduced in Section 1.2.3. We

consider a simplified form of the NLP formulation in (1.9):

$$\min_{X,U,K} \quad \sum_{i=0}^{N-1} l(x_i, u_i) + m(x_N) \tag{6.7a}$$

$$\text{s.t.} \quad 0 = x_0 - \hat{x}_0, \tag{6.7b}$$

$$0 = G(w_i, K_i), \qquad i = 0, \ldots, N-1, \tag{6.7c}$$

$$0 = x_i + B\,K_i - x_{i+1}, \quad i = 0, \ldots, N-1, \tag{6.7d}$$

where $w_i := (x_i, u_i)$ and $z_i := (w_i, K_i)$ and all optimization variables can be concatenated into one vector

$$Z^\top := (x_0, u_0, K_0, \ldots, \underbrace{\underbrace{x_i, u_i}_{w_i}, K_i}_{z_i}, x_{i+1}, u_{i+1}, K_{i+1}, \ldots, x_N) \in \mathbb{R}^{n_\mathrm{Z}}, \tag{6.8}$$

for which $n_\mathrm{Z} = n_\mathrm{W} + N n_\mathrm{K} = n_\mathrm{x} + N(n_\mathrm{x} + n_\mathrm{u} + n_\mathrm{K})$. The Lagrangian for the direct collocation NLP (6.7) is given by

$$\begin{aligned}\mathcal{L}^\mathrm{c}(W, K, \Lambda, \mu) &= \sum_{i=0}^{N-1} \mathcal{L}_i^\mathrm{c}(w_i, K_i, \lambda_i, \mu_i) + m(x_N) \\ &= \lambda_{-1}^\top (x_0 - \hat{x}_0) + \sum_{i=0}^{N-1} \lambda_i^\top (x_i + B\,K_i - x_{i+1}) \\ &\quad + \sum_{i=0}^{N-1} \mu_i^\top G(w_i, K_i) + \sum_{i=0}^{N-1} l(w_i) + m(x_N),\end{aligned} \tag{6.9}$$

where λ_i is defined as before in Eq. (6.6) and μ_i for $i = 0, \ldots, N-1$ denote the multipliers corresponding to the collocation equations (6.7c). For simplicity of notation, we assume in this chapter that the stage cost does not depend on the collocation variables even though there exist optimal control formulations where this function instead reads $\tilde{l}(w_i, K_i)$, e.g., based on continuous output formulas as described in Section 2.5. The local minimizers of the NLPs in (6.4) and (6.7) are assumed to be regular KKT points, corresponding to Definition 1.20.

Remark 6.1 *Based on our expression* (6.5) *for the continuity map* $\phi(x_i, u_i)$ *in Eq.* (6.4c) *defining a fixed step collocation method, both multiple shooting and direct collocation solve the same nonlinear optimization problem. Therefore, a regular KKT point* $(W^\star, K^\star, \Lambda^\star, \mu^\star)$ *to the direct collocation based NLP* (6.7) *forms by definition also a regular KKT point* $(W^\star, \Lambda^\star)$ *to the multiple shooting problem in Eq.* (6.4) *and vice versa.*

6.1.3 Newton-Type Optimization

Based on the discussion in Section 1.3.2, we introduce a Newton-type optimization method for the equality constrained problems in (6.4) and (6.7). In case of direct multiple shooting, a Newton-type scheme iterates by sequentially solving the following linearized system

$$\begin{bmatrix} A & C^\top \\ C & \mathbb{0} \end{bmatrix} \begin{bmatrix} \Delta W \\ \Delta \Lambda \end{bmatrix} = - \begin{bmatrix} a \\ c \end{bmatrix}, \tag{6.10}$$

using the compact notation $\Delta W := (\Delta w_0, \dots, \Delta w_N)$, $w_i := (x_i, u_i)$, $\Delta w_i := w_i - \bar{w}_i$ for $i = 0, \dots, N-1$ and $\Delta w_N := \Delta x_N$. The values $\bar{w}_i := (\bar{x}_i, \bar{u}_i)$ denote the current linearization point instead of the optimization variables w_i and they are updated in each iteration by solving the subproblem (6.10), i.e., $\bar{W}^+ = \bar{W} + \Delta W$ in the case of a full Newton step [232]. The matrices $A \in \mathbb{R}^{n_\mathrm{w} \times n_\mathrm{w}}$, $C \in \mathbb{R}^{(N+1)n_\mathrm{x} \times n_\mathrm{w}}$ are defined as

$$A = \begin{bmatrix} A_0 & & & \\ & \ddots & & \\ & & A_{N-1} & \\ & & & A_N \end{bmatrix}, \; C = \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}}, \mathbb{0} & & & \\ \frac{\partial \phi(\bar{w}_0)}{\partial w_0} & -\mathbb{1}_{n_\mathrm{x}}, \mathbb{0} & & \\ & \ddots & & \\ & & \frac{\partial \phi(\bar{w}_{N-1})}{\partial w_{N-1}} & -\mathbb{1}_{n_\mathrm{x}} \end{bmatrix},$$

in which $C_i := \left[\frac{\partial \phi(\bar{w}_i)}{\partial w_i}, -\mathbb{1}_{n_\mathrm{x}} \right]$ and $A_i := \nabla^2_{w_i} \mathcal{L}_i(\bar{w}_i, \bar{\lambda}_i)$, $A_N := \nabla^2_{x_N} m(\bar{x}_N)$ when using an exact Hessian based Newton method [232]. The Lagrangian term on each shooting interval is thereby defined as $\mathcal{L}_i(\bar{w}_i, \bar{\lambda}_i) = l(\bar{w}_i) + \bar{\lambda}_i^\top (\phi(\bar{w}_i) - \bar{x}_{i+1})$. Note that the initial value condition is included with a term $\bar{\lambda}_{-1}^\top (\bar{x}_0 - \hat{x}_0)$ for the first shooting interval $i = 0$, as in Eq. (6.6). In case of a least squares objective $l(w_i) = \frac{1}{2} \| F(w_i) \|_2^2$, one could alternatively use a Gauss-Newton Hessian approximation such that $A_i := \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top \frac{\partial F(\bar{w}_i)}{\partial w_i}$ [48]. The right-hand side in the KKT system (6.10) consists of $a \in \mathbb{R}^{n_\mathrm{w}}$ and $c \in \mathbb{R}^{(N+1)n_\mathrm{x}}$ defined by

$$a = \begin{bmatrix} a_0 \\ \vdots \\ a_{N-1} \\ a_N \end{bmatrix}, \quad c = \begin{bmatrix} \bar{x}_0 - \hat{x}_0 \\ c_0 \\ \vdots \\ c_{N-1} \end{bmatrix},$$

in which $c_i := \phi(\bar{w}_i) - \bar{x}_{i+1}$ and $a_i := \nabla_{w_i} \mathcal{L}(\bar{W}, \bar{\Lambda})$, $a_N := \nabla_{x_N} \mathcal{L}(\bar{W}, \bar{\Lambda})$.

In a similar fashion, the linearized KKT system can be determined for the direct collocation based NLP (6.7) as

$$\begin{bmatrix} A_\mathrm{c} & E^\top & D^\top \\ E & \mathbb{0} & \mathbb{0} \\ D & \mathbb{0} & \mathbb{0} \end{bmatrix} \begin{bmatrix} \Delta Z \\ \Delta \Lambda \\ \Delta \mu \end{bmatrix} = - \begin{bmatrix} a_\mathrm{c} \\ e \\ d \end{bmatrix}, \tag{6.11}$$

where the matrices $A_{\mathrm{c}} \in \mathbb{R}^{n_{\mathrm{Z}} \times n_{\mathrm{Z}}}$, $D \in \mathbb{R}^{N n_{\mathrm{K}} \times n_{\mathrm{Z}}}$ are block diagonal and defined by $A_{\mathrm{c},i} := \nabla^2_{z_i} \mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i)$ and $D_i := \frac{\partial G(\bar{z}_i)}{\partial z_i}$. In case of a Gauss-Newton Hessian approximation when $l(w_i) = \frac{1}{2}\|F(w_i)\|_2^2$, one has $A_{\mathrm{c},i} := \begin{bmatrix} \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top \frac{\partial F(\bar{w}_i)}{\partial w_i} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \end{bmatrix} \approx \nabla^2_{z_i} \mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i)$ instead. The constant matrix $E \in \mathbb{R}^{(N+1)n_{\mathrm{x}} \times n_{\mathrm{Z}}}$ corresponds to the Jacobian for the continuity constraints (6.7d) and is given by

$$E = \begin{bmatrix} \mathbb{1}_{n_{\mathrm{x}}} & & & & & & & \\ \mathbb{1}_{n_{\mathrm{x}}} & \mathbb{0} & B & -\mathbb{1}_{n_{\mathrm{x}}} & & & & \\ & & & \mathbb{1}_{n_{\mathrm{x}}} & \mathbb{0} & B & -\mathbb{1}_{n_{\mathrm{x}}} & \\ & & & & & \ddots & & \end{bmatrix}. \tag{6.12}$$

The Lagrangian term on each shooting interval now reads as $\mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i) = l(\bar{w}_i) + \bar{\lambda}_i^\top \left(\bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}\right) + \bar{\mu}_i^\top G(\bar{w}_i, \bar{K}_i)$ in Eq. (6.9). The right-hand side components $a_{\mathrm{c}} \in \mathbb{R}^{n_{\mathrm{Z}}}$, $e \in \mathbb{R}^{(N+1)n_{\mathrm{x}}}$ and $d \in \mathbb{R}^{N n_{\mathrm{K}}}$ in the linear system (6.11) can be defined similarly to those of (6.10) where $a_{\mathrm{c},i} := \nabla_{z_i} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu})$, $a_{\mathrm{c},N} := \nabla_{x_N} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu})$, $d_i := G(\bar{w}_i, \bar{K}_i)$ and $e_i := \bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}$.

6.2 Exact Lifted Collocation Integrator

Let us derive the lifted collocation scheme directly from the subproblem in Eq. (6.11), based on a tailored and parallelizable variant of the null space method [232] in order to numerically eliminate the collocation variables. This allows one to further bridge the gap between the Newton-type iterations on the direct collocation and the multiple shooting based problem formulation. Figure 6.2 illustrates this connection and provides an overview of the equations for direct collocation and multiple shooting, both using the standard integrator and with the proposed lifted collocation method.

6.2.1 Structure Exploitation for Direct Collocation

We propose a condensing technique deployed on the Newton step for the direct collocation problem. This allows for the transformation of Eq. (6.11) into the form of (6.10) and thereby application of the tools developed for the multiple shooting approach. We present this result as the following proposition.

Proposition 6.2 *Algorithm 5 solves the linearized direct collocation KKT system in Eq.* (6.11) *by performing a condensing technique, followed by solving*

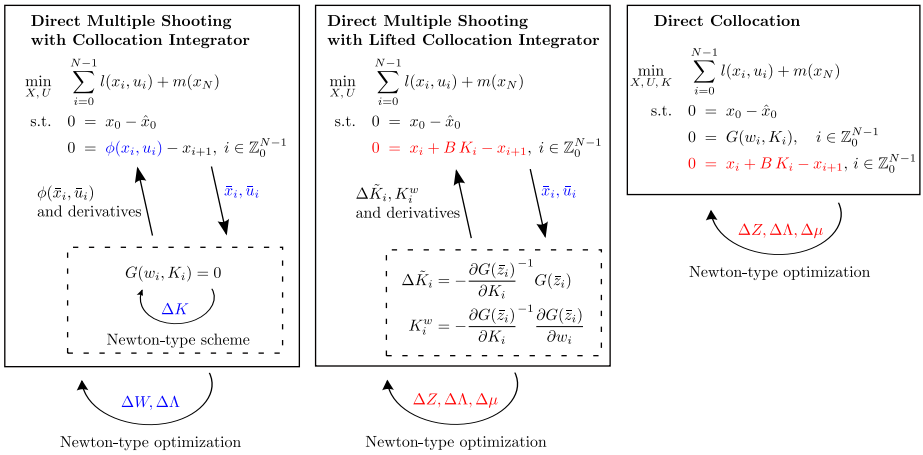


Figure 6.2: An overview of the idea of using lifted collocation integrators, with combined properties from multiple shooting and direct collocation.

a multiple shooting type KKT system of the form (6.10) *and a corresponding expansion procedure to obtain the full solution* $(\Delta Z, \Delta\Lambda, \Delta\mu)$.

Proof. Let us start with the following expressions resulting from the continuity and collocation equations on the second and third line of the direct collocation based KKT system (6.11), i.e.,

$$\frac{\partial G(\bar{z}_i)}{\partial w_i}\Delta w_i + \frac{\partial G(\bar{z}_i)}{\partial K_i}\Delta K_i = -d_i \qquad \text{and}$$

$$\Delta x_i + B\,\Delta K_i - \Delta x_{i+1} = -e_i,$$

for each $i = 0, \dots, N-1$, where the previous definition of the matrices D_i and E has been used and, additionally, $d_i = G(\bar{z}_i)$ and $e_i = \bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}$. Since the Jacobian $\frac{\partial G(\bar{z}_i)}{\partial K_i}$ is nonsingular [158], one can eliminate the collocation variables $\Delta K_i = \Delta\tilde{K}_i + K_i^w \Delta w_i$ from the subsystem which reads as

$$\Delta x_i + B\,K_i^w \Delta w_i - \Delta x_{i+1} = -\tilde{e}_i,$$

where $\tilde{e}_i := e_i + B\,\Delta\tilde{K}_i$ and the auxiliary variables

$$\begin{aligned} \Delta\tilde{K}_i &= -\frac{\partial G(\bar{z}_i)}{\partial K_i}^{-1} G(\bar{z}_i) \qquad \text{and} \\ K_i^w &= -\frac{\partial G(\bar{z}_i)}{\partial K_i}^{-1} \frac{\partial G(\bar{z}_i)}{\partial w_i} \end{aligned} \tag{6.13}$$

have been defined. Subsequently, we look at the first line of the direct collocation based KKT system (6.11),

$$\underbrace{\nabla^2_{z_i}\mathcal{L}^{\mathrm{c}}_i}_{=A_{\mathrm{c},i}}\Delta z_i + E_i^\top \Delta\lambda_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}}\\ \mathbb{0}\\ \mathbb{0}\end{bmatrix}\Delta\lambda_{i-1} + \underbrace{\frac{\partial G(\bar{z}_i)}{\partial z_i}^\top}_{=D_i^\top}\Delta\mu_i = -\underbrace{\nabla_{z_i}\mathcal{L}^{\mathrm{c}}}_{=a_{\mathrm{c},i}}, \tag{6.14}$$

where the matrix $E_i = \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} & \mathbb{0} & B\end{bmatrix}$ has been defined. Since $\Delta K_i = \Delta\tilde{K}_i + K_i^w \Delta w_i$, we may write $\Delta z_i = \begin{bmatrix}\Delta w_i\\ \Delta K_i\end{bmatrix} = \begin{bmatrix}\mathbb{1}_{n_\mathrm{w}}\\ K_i^w\end{bmatrix}\Delta w_i + \begin{bmatrix}\mathbb{0}\\ \mathbb{1}_{n_\mathrm{K}}\end{bmatrix}\Delta\tilde{K}_i$ which, when applied to (6.14), yields

$$\begin{aligned}\left(\nabla^2_{z_i,w_i}\mathcal{L}^{\mathrm{c}}_i + \nabla^2_{z_i,K_i}\mathcal{L}^{\mathrm{c}}_i K_i^w\right)\Delta w_i + E_i^\top\Delta\lambda_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}}\\ \mathbb{0}\\ \mathbb{0}\end{bmatrix}\Delta\lambda_{i-1} + \frac{\partial G(\bar{z}_i)}{\partial z_i}^\top\Delta\mu_i\\ = -\nabla_{z_i}\mathcal{L}^{\mathrm{c}} - \nabla^2_{z_i,K_i}\mathcal{L}^{\mathrm{c}}_i\Delta\tilde{K}_i.\end{aligned} \tag{6.15}$$

Additionally, we observe that

$$\begin{aligned}\frac{\partial G(\bar{z}_i)}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}w_i} &= \frac{\partial G(\bar{z}_i)}{\partial w_i} + \frac{\partial G(\bar{z}_i)}{\partial K_i}K_i^w\\ &= \frac{\partial G(\bar{z}_i)}{\partial w_i} - \frac{\partial G(\bar{z}_i)}{\partial K_i}\frac{\partial G(\bar{z}_i)}{\partial K_i}^{-1}\frac{\partial G(\bar{z}_i)}{\partial w_i} = \mathbb{0},\end{aligned}$$

where $\frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top = \begin{bmatrix}\mathbb{1}_{n_\mathrm{w}} & K_i^{w^\top}\end{bmatrix}$. This can be used to simplify Eq. (6.15). Left multiplying both sides of (6.15) with $\frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top$ results in

$$A_i\Delta w_i + \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} + K_i^{x^\top}B^\top\\ K_i^{u^\top}B^\top\end{bmatrix}\Delta\lambda_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}}\\ \mathbb{0}\end{bmatrix}\Delta\lambda_{i-1} = -a_i,$$

where the Hessian matrix can be written as

$$\begin{aligned}A_i &= \left(\nabla^2_{w_i}\mathcal{L}^{\mathrm{c}}_i + K_i^{w^\top}\nabla^2_{K_i,w_i}\mathcal{L}^{\mathrm{c}}_i + \nabla^2_{w_i,K_i}\mathcal{L}^{\mathrm{c}}_i K_i^w + K_i^{w^\top}\nabla^2_{K_i}\mathcal{L}^{\mathrm{c}}_i K_i^w\right)\\ &= \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top\nabla^2_{z_i}l(\bar{w}_i)\frac{\mathrm{d}z_i}{\mathrm{d}w_i} + \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top\langle\bar{\mu}_i,\nabla^2_{z_i}G_i\rangle\frac{\mathrm{d}z_i}{\mathrm{d}w_i}\\ &= \nabla^2_{w_i}l(\bar{w}_i) + H_i,\end{aligned} \tag{6.16}$$

in which $H_i := \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top\langle\bar{\mu}_i,\nabla^2_{z_i}G_i\rangle\frac{\mathrm{d}z_i}{\mathrm{d}w_i}$ is the condensed Hessian contribution from the collocation equations. Here, the notation $\langle\bar{\mu},\nabla^2_z G\rangle = \sum_{r=1}^{n_\mathrm{K}}\bar{\mu}_r\frac{\partial^2 G_r}{\partial z^2}$ is used.

The right-hand side reads as

$$\begin{aligned} a_i &= \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \nabla_{z_i} \mathcal{L}^{\mathrm{c}} + \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \nabla^2_{z_i,K_i} \mathcal{L}^{\mathrm{c}}_i \Delta \tilde{K}_i \\ &= \nabla_{w_i} \mathcal{L}^{\mathrm{c}} + K_i^{w^\top} \nabla_{K_i} \mathcal{L}^{\mathrm{c}} + \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i,K_i} G_i \rangle \, \Delta \tilde{K}_i \\ &= \nabla_{w_i} l(\bar{w}_i) + \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} + K_i^{x^\top} B^\top \\ K_i^{u^\top} B^\top \end{bmatrix} \bar{\lambda}_i - \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0} \end{bmatrix} \bar{\lambda}_{i-1} + h_i, \end{aligned} \tag{6.17}$$

where we used $\frac{\partial G(\bar{z}_i)}{\partial z_i} \frac{\mathrm{d}z_i}{\mathrm{d}w_i} = 0$ and $h_i := \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i,K_i} G_i \rangle \, \Delta \tilde{K}_i$.

Based on the numerical elimination or *condensing* of the collocation variables ΔK_i, the KKT system from Eq. (6.11) can be rewritten in the multiple-shooting form of Eq. (6.10) where the matrices C and A are defined by

$$C_i = \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} + B\, K_i^x & B\, K_i^u & -\mathbb{1}_{n_\mathrm{x}} \end{bmatrix}, \quad A_i = \nabla^2_{w_i} l(\bar{w}_i) + H_i, \tag{6.18}$$

respectively. The vectors c and a on the right-hand side of the system are defined by

$$c_i = \tilde{e}_i, \quad a_i = \nabla_{w_i} l(\bar{w}_i) + \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} + K_i^{x^\top} B^\top \\ K_i^{u^\top} B^\top \end{bmatrix} \bar{\lambda}_i - \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0} \end{bmatrix} \bar{\lambda}_{i-1} + h_i, \tag{6.19}$$

for each $i = 0, \ldots, N-1$. After solving the resulting multiple shooting type KKT system (6.10), one can obtain the full direct collocation solution by performing the following *expansion* step for the lifted variables K and μ:

$$\begin{aligned} \Delta K_i &= \Delta \tilde{K}_i + K_i^w \Delta w_i \\ \mu_i^+ &= -\frac{\partial G_i}{\partial K_i}^{-\top} \left(B^\top \lambda_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i,z_i} G_i \rangle \, \Delta z_i \right), \end{aligned} \tag{6.20}$$

using the Newton step $(\Delta W, \Delta \Lambda)$ where $\lambda_i^+ = \bar{\lambda}_i + \Delta \lambda_i$. The expansion step (6.20) for the Lagrange multipliers μ_i can be found by looking at the lower part of the KKT conditions in Eq. (6.14),

$$\nabla^2_{K_i,z_i} \mathcal{L}^{\mathrm{c}}_i \Delta z_i + B^\top \Delta \lambda_i + \frac{\partial G_i}{\partial K_i}^\top \Delta \mu_i = -\nabla_{K_i} \mathcal{L}^{\mathrm{c}},$$

which can be rewritten as

$$\frac{\partial G_i}{\partial K_i}^\top \Delta \mu_i = -\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i - B^\top \bar{\lambda}_i - B^\top \Delta \lambda_i - \langle \bar{\mu}_i, \nabla^2_{K_i,z_i} G_i \rangle \, \Delta z_i. \tag{6.21}$$

□

Algorithm 5 Newton-type optimization step, based on the exact lifted collocation integrator within direct multiple shooting (LC-EN).

Input: Current values $\bar{z}_i = (\bar{x}_i, \bar{u}_i, \bar{K}_i)$ and $(\bar{\lambda}_i, \bar{\mu}_i)$ for $i = 0, \dots, N-1$.
Output: Updated values $\bar{z}_i^+$ and $(\bar{\lambda}_i^+, \bar{\mu}_i^+)$ for $i = 0, \dots, N-1$.

`Condensing procedure`

1: **for** $i = 0, \dots, N-1$ **do in parallel** (forward sweep)

2: Compute the values $\Delta\tilde{K}_i$ and K_i^w using Eq. (6.13):
$\Delta\tilde{K}_i \leftarrow -\frac{\partial G_i}{\partial K_i}^{-1} G(\bar{z}_i)$ and $K_i^w \leftarrow -\frac{\partial G_i}{\partial K_i}^{-1} \frac{\partial G_i}{\partial w_i}$.

3: Hessian and gradient terms using (6.16)-(6.17):
$H_i \leftarrow \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle \frac{\mathrm{d}z_i}{\mathrm{d}w_i}$ and $h_i \leftarrow \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \Delta\tilde{K}_i$.

4: **end for**

`Computation of step direction`

5: Solve the linear system (6.10) based on the data C_i, A_i and c_i, a_i in (6.18) and (6.19) for $i = 0, \dots, N-1$, in order to obtain the step $(\Delta W, \Delta\Lambda)$.
$\bar{w}_i^+ \leftarrow \bar{w}_i + \Delta w_i$ and $\bar{\lambda}_i^+ \leftarrow \bar{\lambda}_i + \Delta\lambda_i$.

`Expansion procedure`

6: **for** $i = 0, \dots, N-1$ **do in parallel** (backward sweep)

7: The full solution can be obtained using Eq. (6.20):
$\bar{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + K_i^w \Delta w_i$.
$\bar{\mu}_i^+ \leftarrow -\frac{\partial G_i}{\partial K_i}^{-\top} \left(B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \Delta z_i \right)$.

8: **end for**

Remark 6.3 *Algorithm 5 can be readily extended to nonlinear inequality constrained optimization, since the lifted collocation integrator is not directly affected by such inequality constraints. More specifically, the presence of inequality constraints only influences the computation of the step direction based on the KKT conditions [232]. Therefore, the lifted collocation scheme can, for example, be implemented within an SQP method [52] by linearizing the inequality constraints and solving the resulting QP subproblem to compute the step direction in Algorithm 5. Note that such an SQP type implementation is performed in the* `ACADO Toolkit` *as discussed later in Section 6.5. Similarly, an IP method [39] could be implemented based on the lifted collocation integrator so that the step direction computation in Algorithm 5 involves the solution of the (primal-dual) interior point system.*

Remark 6.4 *Proposition 6.2 presents a specific condensing and expansion technique which can also be interpreted as a parallelizable linear algebra routine to exploit the specific direct collocation structure in the Newton method, instead of relying on general-purpose sparse linear algebra packages. A similar idea of*

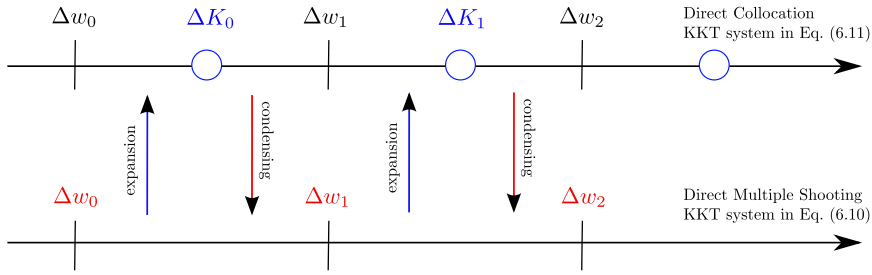


Figure 6.3: Illustration of the parallelizable condensing and expansion to efficiently eliminate and recover the collocation variables from the linearized KKT system.

using specialized linear algebra to solve the KKT system for direct collocation has been proposed in [186, 326, 334], based on interior point methods and Schur complement techniques. Note that the numerical elimination of the collocation variables by computing the corresponding quantities in Eqs. (6.18) *and* (6.19) *can be performed independently and therefore in parallel for each shooting interval* $i = 0, \dots, N-1$ *as illustrated by Figure 6.3. The same holds true for the expansion step in Eq.* (6.20) *to recover the full solution.*

6.2.2 The Lifted Collocation Algorithm

Algorithm 5 presents the exact *lifted collocation* scheme (LC-EN), which can be used within direct multiple shooting based on the results of Proposition 6.2. The resulting Newton-type optimization algorithm takes steps $(\Delta W, \Delta K, \Delta\Lambda, \Delta\mu)$ that are equivalent to those for Newton-type optimization applied to the direct collocation based NLP. Given a regular KKT point, $(W^\star, K^\star, \Lambda^\star, \mu^\star)$, as in Definition 1.20 for this NLP (6.7), the lifted collocation algorithm therefore converges with a linear rate to this minimizer in the case of a Gauss-Newton Hessian approximation or with a locally quadratic convergence rate in the case of an exact Hessian method [232]. Note that more recent results on inexact Newton-type optimization algorithms exist, e.g., allowing locally superlinear [94] or even quadratic convergence rates [175] under certain conditions.

Connection to the standard lifted Newton method

The lifted Newton method [8] identifies intermediate values in the constraints and objective functions and introduces them as additional degrees of freedom in the NLP. Instead of solving the resulting equivalent (but higher dimensional)

optimization problem directly, a condensing and expansion step are proposed to give a computational burden similar to the non-lifted Newton type optimization algorithm. The present chapter proposes an extension of that concept to intermediate variables that are instead defined implicitly, such as the collocation variables on each shooting interval. Similar to the discussion for the lifted Newton method in [8], the lifted collocation integrator offers multiple advantages over the non-lifted method such as an improved local convergence. Most important, unlike the standard lifted Newton method, the lifting of implicitly defined variables avoids the need for an iterative scheme within each iteration of the Newton-type optimization algorithm, and therefore typically reduces the computational effort. These properties will be detailed next.

Comparison with direct collocation and multiple shooting

This section compares multiple shooting (MS), lifted collocation (LC) and direct collocation (DC), all aimed at solving the same nonlinear optimization problem in Eq. (6.7) (see Remark 6.1). Proposition 6.2 shows that lifted and direct collocation result in the exact same Newton-type iterations and therefore share the same convergence properties. The arguments proposed in [8] for the lifted Newton method suggest that this local convergence can be better than for direct multiple shooting based on a collocation method. However, the main motivation for using lifting in this chapter is that, internally, multiple shooting requires Newton-type iterations to solve the collocation equations (6.2) within each NLP iteration to evaluate the continuity map while lifted collocation avoids such internal iterations. In addition, let us mention some of the advantages of lifted collocation over the use of direct collocation:

- The elimination of the collocation variables, i.e., the *condensing*, can be performed in a tailored, structure-exploiting manner. Similarly to direct multiple shooting, the proposed condensing technique can be highly and straightforwardly parallelized since the elimination of the variables ΔK_i on each shooting interval can be done independently.
- The resulting condensed subproblem is smaller but still sparse, since it is of the multiple-shooting form (6.10). It therefore offers the additional practical advantage that one can deploy any of the embedded solvers tailored for the multi-stage quadratic subproblem with a specific optimal control structure, such as `FORCES` [96], `qpDUNES` [117] or `HPMPC` [123].
- An important advantage of multiple shooting over direct collocation is the possibility of using any ODE or DAE solver, including step size and order control to guarantee a specific integration accuracy [51, 158]. Such

Table 6.1: Comparison of the three collocation based approaches to solve the nonlinear optimal control problem in Eq. (6.7).

	Multiple Shooting (MS)	Lifted Collocation (LC)	Direct Collocation (DC)
Step size control	+	0	0
Embedded solvers	+	+	-
Parallelizability	+	+	0
Local convergence	0	+	+
Internal iterations	-	+	+
Sparsity dynamics	-	-	+

an adaptive approach becomes more difficult, but can be combined with direct collocation where the problem dimensions change in terms of the step size and order of the polynomial [36, 215, 246]. Even though it is out of the scope of this work, the presented lifting technique allows one to implement similar approaches while keeping the collocation variables hidden from the NLP solver based on condensing and expansion.

- The main advantage of direct collocation over multiple shooting is the better preservation of sparsity in the derivative matrices. Additionally, the evaluation of derivatives for the collocation equations is typically cheaper than the propagation of sensitivities for an integration scheme.

The above observations are summarized in Table 6.1, which lists advantages and disadvantages for all three approaches. It is important to note that direct collocation is also highly parallelizable, although one needs to rely on an advanced linear algebra package for detecting the sparsity structure of Eq. (6.11), exploiting it and performing the parallelization. In contrast, the lifted collocation approach is parallelizable in a natural way and independently of the chosen linear algebra. The relative performance of using a general-purpose sparse linear algebra routine for direct collocation versus the proposed approach depends very much on the specific problem dimensions and structure, and on the solver used. It has been shown in specific contexts that structure exploiting implementations of optimal control methods based on dense linear algebra routines typically outperform general-purpose solvers [121]. This topic will be discussed further for direct collocation in the numerical case study of Section 6.6.

6.2.3 Forward-Backward Propagation

The efficient computation of second-order derivatives using Algorithmic Differentiation (AD) is typically based on a forward sweep, followed by a backward propagation of the derivatives as detailed in [141]. Inspired by this approach, Algorithm 5 proposes to perform the condensing and expansion step using such a forward-backward propagation in the case of exact Hessian based optimization. To reveal these forward and backward propagation sweeps in Algorithm 5 explicitly, we recall the structure of the collocation equations from the formulation in (6.2), where we omit the shooting index, $i = 0, \dots, N-1$, to obtain the compact notation

$$G(w,K) = \begin{bmatrix} g_1(w,K_1) \\ \vdots \\ g_{N_\mathrm{s}}(w,K_1,\dots,K_{N_\mathrm{s}}) \end{bmatrix} = \begin{bmatrix} g_1(w_0,K_1) \\ \vdots \\ g_{N_\mathrm{s}}(w_{N_\mathrm{s}-1},K_{N_\mathrm{s}}) \end{bmatrix} = 0. \tag{6.22}$$

Here, $w_0 = (x,u)$, $w_n = (x_n,u)$ and $x_n = x_{n-1} + B_n K_n$ denotes the intermediate state values in Eq. (6.3) such that the numerical simulation result $\phi(w) = x_{N_\mathrm{s}}$ is defined. Let us briefly present the forward-backward propagation scheme for respectively the *condensing* and *expansion* step of Algorithm 5 within one shooting interval.

Condensing the lifted variables: forward sweep

The condensing procedure in Algorithm 5 aims to compute the data $C = \left[\frac{\mathrm{d}x_{N_\mathrm{s}}}{\mathrm{d}w_0}, -\mathbb{1}_{n_\mathrm{x}}\right]$ and $A = \nabla_w^2 l(w) + H$, where the matrix $H = \frac{\mathrm{d}z}{\mathrm{d}w}^\top \langle \mu, \nabla_z^2 G\rangle \frac{\mathrm{d}z}{\mathrm{d}w}$ is defined similar to Eq. (6.16). In addition, the vectors $c = e_i + B\,\Delta\tilde{K}$ and $a = \nabla_w l(w) + \frac{\mathrm{d}x_{N_\mathrm{s}}}{\mathrm{d}w_0}^\top \lambda_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0}\end{bmatrix} \lambda_{i-1} + h$, in which $h = \frac{\mathrm{d}z}{\mathrm{d}w}^\top \langle \mu, \nabla_{z,K}^2 G\rangle\, \Delta\tilde{K}$, are needed to form the linearized multiple shooting type KKT system (6.10). Note that this forms a simplified formulation of the condensed expressions in Eqs. (6.18) and (6.19) within one shooting interval.

Given the particular structure of the collocation equations in (6.22) for N_s integration steps, the variables K_n can be eliminated sequentially for $n = 1, \dots, N_\mathrm{s}$. The lifted Newton step $\Delta\tilde{K} = -\frac{\partial G}{\partial K}^{-1} G(\bar{z})$ can therefore be written as the following forward sequence

$$\Delta\tilde{K}_n = -\frac{\partial g_n}{\partial K_n}^{-1} \left(g_n + \frac{\partial g_n}{\partial x_{n-1}} \Delta\tilde{x}_{n-1} \right), \tag{6.23}$$

for $n = 1, \dots, N_\mathrm{s}$ and where $g_n := g_n(\bar{w}_{n-1}, \bar{K}_n)$ and $\Delta\tilde{x}_0 = 0$ so that $\Delta\tilde{x}_n = \Delta\tilde{x}_{n-1} + B_n\,\Delta\tilde{K}_n$. The same holds for the corresponding first order forward

sensitivities $K^w = -\frac{\partial G}{\partial K}^{-1} \frac{\partial G}{\partial w}$, which read as

$$K_n^w := \frac{\mathrm{d}K_n}{\mathrm{d}w_0} = -\frac{\partial g_n}{\partial K_n}^{-1} \left(\frac{\partial g_n}{\partial w_{n-1}} \frac{\mathrm{d}w_{n-1}}{\mathrm{d}w_0} \right), \tag{6.24}$$

where the first order derivatives $\frac{\mathrm{d}w_{n-1}}{\mathrm{d}w_0} = \begin{bmatrix} S_{n-1} \\ \mathbb{0} \; \mathbb{1}_{n_\mathrm{u}} \end{bmatrix}$ and $S_n = \frac{\mathrm{d}x_n}{\mathrm{d}w_0}$ are defined. These sensitivities are used to propagate the state derivatives

$$S_n = S_{n-1} + B_n \, K_n^w \tag{6.25}$$

for $n = 1, \ldots, N_\mathrm{s}$. This forward sequence, starting at $S_0 = \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} & \mathbb{0} \end{bmatrix}$, results in the complete Jacobian $S_{N_\mathrm{s}} = \frac{\mathrm{d}x_{N_\mathrm{s}}}{\mathrm{d}w_0}$.

After introducing the compact notation $\mu_n^\top g_n(\bar{w}_{n-1}, \bar{K}_n) = \sum_{r=1}^q \mu_{n,r}^\top f_{n,r}$ where $f_{n,r} := f(\bar{k}_{n,r}, \bar{w}_{n,r})$ denote the dynamic function evaluations in (6.2), the expressions for the second-order sensitivities are

$$K_n^{w,w} = \sum_{r=1}^{q} \frac{\mathrm{d}z_{n,r}}{\mathrm{d}w_0}^\top \langle \mu_{n,r}, \nabla^2_{z_{n,r}} f_{n,r} \rangle \frac{\mathrm{d}z_{n,r}}{\mathrm{d}w_0}, \tag{6.26}$$

where $z_{n,r} := (k_{n,r}, w_{n,r})$, $w_{n,r} := (x_{n,r}, u)$ and the stage values are defined by $x_{n,r} = x_{n-1} + T_\mathrm{int} \sum_{s=1}^q a_{r,s} k_{n,s}$. The derivatives $\frac{\mathrm{d}z_{n,r}}{\mathrm{d}w_0}$ are based on the first-order forward sensitivity information in Eqs. (6.24) and (6.25). In a similar way to that described in Chapter 3, one can additionally perform a forward symmetric Hessian propagation sweep,

$$H_n = H_{n-1} + K_n^{w,w}, \tag{6.27}$$

for $n = 1, \ldots, N_\mathrm{s}$ and $H_0 = \mathbb{0}$ such that $H_{N_\mathrm{s}} = \sum_{n=1}^{N_\mathrm{s}} K_n^{w,w}$. Regarding the gradient contribution, one can propagate the following sequence

$$h_n = h_{n-1} + \sum_{r=1}^{q} \frac{\mathrm{d}z_{n,r}}{\mathrm{d}w_0}^\top \langle \mu_{n,r}, \nabla^2_{z_{n,r}} f_{n,r} \rangle \Delta \tilde{z}_{n,r}, \tag{6.28}$$

for $n = 1, \ldots, N_\mathrm{s}$, where the values $\Delta \tilde{x}_{n,r} = \Delta \tilde{x}_{n-1} + T_\mathrm{int} \sum_{s=1}^q a_{r,s} \Delta \tilde{k}_{n,s}$ are defined. Given the initial values $H_0 = \mathbb{0}$ and $h_0 = \mathbb{0}$, the forward sweeps (6.27)-(6.28) result in $H_{N_\mathrm{s}} = \frac{\mathrm{d}z}{\mathrm{d}w}^\top \langle \bar{\mu}, \nabla_z^2 G \rangle \frac{\mathrm{d}z}{\mathrm{d}w}$ and $h_{N_\mathrm{s}} = \frac{\mathrm{d}z}{\mathrm{d}w}^\top \langle \bar{\mu}, \nabla^2_{z,K} G \rangle \, \Delta \tilde{K}$.

Remark 6.5 *The above computations to evaluate the condensed Hessian contribution shows a resemblance with the classical condensing method to eliminate the state variables in direct optimal control [51]. The main difference is that the above condensing procedure is carried out independently for the state and control variable within each shooting interval, such that the number of optimization variables does not increase in this case.*

Expansion step for the lifted variables: backward sweep

Note that the first and second order sensitivities can be propagated together in the forward condensing scheme, which avoids unnecessary additional storage requirements. We show next that the expansion phase of Algorithm 5 can be seen as the subsequent backward propagation sweep. For this purpose, certain variables from the forward scheme still need to be stored.

The expansion step $\bar{K}^+ = \bar{K} + \Delta\tilde{K} + K^w \Delta w$ for the lifted collocation variables can be performed as follows

$$\bar{K}_n^+ = \bar{K}_n + \Delta\tilde{K}_n + K_n^w \Delta w_0 \quad \text{for} \quad n = 1, \ldots, N_\mathrm{s}, \tag{6.29}$$

where the values $\Delta\tilde{K}_n$ and K_n^w are obtained from the condensing procedure and Δw_0 denotes the primal update from the subproblem solution in Algorithm 5. The expansion step $\bar{\mu}^+ = -\frac{\partial G}{\partial K}^{-\top} \left(B^\top \bar{\lambda}^+ + \langle \bar{\mu}, \nabla^2_{K,z} G \rangle \, \Delta z \right)$ for the lifted dual variables can be performed as a backward propagation

$$\begin{aligned} \bar{\mu}_n^+ &= -\frac{\partial g_n}{\partial K_n}^{-\top} \left(B_n^\top \bar{\lambda}_n^+ + \sum_{m=n}^{N_\mathrm{s}} \langle \bar{\mu}_m, \nabla^2_{K_n, z_m} g_m \rangle \Delta z_m \right), \\ \text{where} \quad \bar{\lambda}_{n-1}^+ &= \bar{\lambda}_n^+ + \frac{\partial g_n}{\partial x_{n-1}}^\top \bar{\mu}_n^+, \end{aligned} \tag{6.30}$$

for $n = N_\mathrm{s}, \ldots, 1$, based on the initial value $\bar{\lambda}_{N_\mathrm{s}}^+ = \bar{\lambda}^+$ from the subproblem solution, and where $\Delta x_n = \Delta x_{n-1} + B_n \Delta K_n$ and $\Delta z_n = (\Delta w_n, \Delta K_n)$. Note that the factorization of the Jacobian $\frac{\partial g_n}{\partial K_n}$ is needed from the forward propagation to efficiently perform this backward sweep.

6.2.4 Lifted Collocation Integrator with Gauss-Newton

The previous subsection detailed how the expressions in Algorithm 5 can be computed by a forward-backward propagation which exploits the symmetry of the exact Hessian contribution. In the case when a Gauss-Newton or Quasi-Newton type optimization method is used, the Hessian contribution from the dynamic constraints is $H_i = 0$ and the gradient $h_i = 0$ for $i = 0, \ldots, N-1$, since no second-order derivative propagation is needed. The multipliers μ corresponding to the collocation equations are then not needed either, so that only the collocation variables K are lifted. In this context, Algorithm 5 boils down to a forward sweep for both the condensing and the expansion steps of the scheme without the need for additional storage of intermediate values, unlike the case of the forward-backward propagation.

6.3 Adjoint-based Inexact Lifted Collocation

Any implementation of a collocation method needs to compute the collocation variables K_i from the nonlinear equations $G(w_i, K_i) = 0$, given the current values for w_i. The earlier definition of the auxiliary variable $\Delta\tilde{K}_i$ in Eq. (6.13) corresponds to an exact Newton step $\Delta\tilde{K}_i = -\frac{\partial G(\bar{w}_i,\bar{K}_i)}{\partial K_i}^{-1} G(\bar{w}_i, \bar{K}_i)$. It is, however, common in practical implementations of collocation methods or IRK schemes in general to use inexact derivative information to approximate the Jacobian matrix, $M_i \approx \frac{\partial G(\bar{w}_i,\bar{K}_i)}{\partial K_i}$, resulting in the inexact Newton step

$$\Delta\tilde{K}_i = -M_i^{-1} G(\bar{w}_i, \bar{K}_i). \tag{6.31}$$

This Jacobian approximation can allow for a computationally cheaper LU factorization, which can be reused throughout the iterations [158]. Monitoring strategies on when to reuse such a Jacobian approximation is a research topic of its own, e.g., see [7, 27]. Additionally, there exist iterative ways of updating the Jacobian approximation, e.g., based on Broyden's method [57]. Efficient implementations of IRK methods based on such a tailored Jacobian approximation M_i, are, for example, known as the *Simplified* [35, 62] and the *Single Newton* iteration [71, 138] as presented earlier in Section 2.4.

6.3.1 Adjoint-based Inexact Lifting Algorithm

Even though it can be computationally attractive to use the inexact Newton scheme from Eq. (6.31) instead of the exact method, its impact on the convergence of the resulting Newton-type optimization algorithm is an important topic which is addressed in more detail by [44, 82, 94, 252]. A Newton-type scheme with inexact derivatives does not converge to a solution of the original direct collocation NLP (6.7), unless adjoint derivatives are evaluated in order to compute the correct gradient of the Lagrangian $a_{\mathrm{c},i} = \nabla_{z_i} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu})$ on the right-hand side of the KKT system (6.11) [49, 94].

Let us introduce the Jacobian approximation $\tilde{D}_i = [\frac{\partial G(\bar{z}_i)}{\partial w_i}, M_i] \approx \frac{\partial G(\bar{z}_i)}{\partial z_i} \in \mathbb{R}^{n_{\mathrm{K}} \times n_z}$, where $M_i \approx \frac{\partial G(\bar{z}_i)}{\partial K_i}$ is invertible for each $i = 0, \ldots, N-1$, and which is possibly fixed. One then obtains the inexact KKT system

$$\begin{bmatrix} A_{\mathrm{c}} & E^\top & \tilde{D}^\top \\ E & \mathbb{0} & \mathbb{0} \\ \tilde{D} & \mathbb{0} & \mathbb{0} \end{bmatrix} \begin{bmatrix} \Delta Z \\ \Delta\Lambda \\ \Delta\mu \end{bmatrix} = - \begin{bmatrix} a_{\mathrm{c}} \\ e \\ d \end{bmatrix}, \tag{6.32}$$

where all matrices and vectors are defined as for the direct collocation based KKT system in Eq. (6.11), with the exception of $\tilde{D}$, where the Jacobian

approximations M_i are used instead of $\frac{\partial G(\bar{z}_i)}{\partial K_i}$. The resulting iterative scheme is known as an adjoint-based inexact Newton method [49, 94] applied to the direct collocation NLP in Eq. (6.7) because the right-hand side is evaluated exactly, including the gradient of the Lagrangian, $a_{\mathrm{c},i} = \nabla_{z_i} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu})$. We detail this approach in Algorithm 6 and motivate it by the following proposition.

Proposition 6.6 *Algorithm 6 presents a condensing technique for the inexact KKT system* (6.32)*, which allows one to instead solve a system of the multiple-shooting form in Eq.* (6.10)*. The solution* $(\Delta Z, \Delta\Lambda, \Delta\mu)$ *to the original system* (6.32) *can be obtained by use of the corresponding expansion technique.*

Proof. The proof here follows similar arguments as that used for Proposition 6.2, with the main difference that the update of the collocation variables is instead given by $\Delta K_i = \Delta\tilde{K}_i + \tilde{K}_i^w \Delta w_i$, where

$$\Delta\tilde{K}_i = -M_i^{-1} G(\bar{z}_i), \quad \tilde{K}_i^w = -M_i^{-1} \frac{\partial G(\bar{z}_i)}{\partial w_i}, \tag{6.33}$$

and where $\tilde{K}_i^w$ denotes the inexact forward sensitivities. To obtain the multiple shooting type form of the KKT system in Eq. (6.10), the resulting condensing and expansion step can be found in Algorithm 6. An important difference with the exact lifted collocation integrator from Algorithm 5 is that the gradient term h_i is now defined as

$$h_i = z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \, \Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)^\top \bar{\mu}_i, \tag{6.34}$$

where $z_i^{w\top} := \begin{bmatrix} \mathbb{1}_{n_\mathrm{w}} & \tilde{K}_i^{w\top} \end{bmatrix}$ and including a correction term resulting from the inexact sensitivities $\tilde{K}_i^w$. In addition, the expansion step for the Lagrange multipliers corresponding to the collocation equations is now

$$\Delta\mu_i = -M_i^{-\top} \left(\frac{\partial G(\bar{z}_i)}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \, \Delta z_i \right), \tag{6.35}$$

which corresponds to an inexact Newton-type iteration on the exact Newton based expression from Eq. (6.21).

□

Table 6.2 shows an overview of the presented variants of lifted collocation including the exact method (LC-EN) in Algorithm 5, which can be compared to the adjoint based inexact lifting scheme (LC-IN) in Algorithm 6.

Exact Lifted Collocation (LC-EN) **Algorithm 5**	Adjoint-based Inexact Lifting (LC-IN) **Algorithm 6**	Inexact Lifting with Iterated Sensitivities (LC-INIS) **Algorithm 7-8**
Condensing procedure for $i = 0, \dots, N-1$ (forward sweep)		
$\Delta\tilde{K}_i \leftarrow -\frac{\partial G_i}{\partial K_i}^{-1} G(\bar{z}_i)$ $K_i^w \leftarrow -\frac{\partial G_i}{\partial K_i}^{-1} \frac{\partial G_i}{\partial w_i}$	$\Delta\tilde{K}_i \leftarrow -M_i^{-1} G(\bar{z}_i)$ $\tilde{K}_i^w \leftarrow -M_i^{-1} \frac{\partial G_i}{\partial w_i}$	$\Delta\tilde{K}_i \leftarrow -M_i^{-1} G(\bar{z}_i)$ $\Delta K_i^w \leftarrow -M_i^{-1} \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)$
(Exact Hessian):	**(Exact Hessian)**:	**(Exact Hessian)**:
$H_i \leftarrow \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle \frac{\mathrm{d}z_i}{\mathrm{d}w_i}$ $h_i \leftarrow \frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \Delta\tilde{K}_i$ where $\frac{\mathrm{d}z_i}{\mathrm{d}w_i}^\top = \begin{bmatrix} \mathbb{1}_{n_\mathrm{w}} & K_i^{w\top} \end{bmatrix}$	$H_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle z_i^w$ $h_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)^\top \bar{\mu}_i$ where $z_i^{w\top} = \begin{bmatrix} \mathbb{1}_{n_\mathrm{w}} & \tilde{K}_i^{w\top} \end{bmatrix}$	$H_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle z_i^w$ $h_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)^\top \bar{\mu}_i$ where $z_i^{w\top} = \begin{bmatrix} \mathbb{1}_{n_\mathrm{w}} & \tilde{K}_i^{w\top} \end{bmatrix}$
(Gauss-Newton):	**(Gauss-Newton)**:	**(Gauss-Newton)**:
-	$H_i \leftarrow \mathbb{0}$ and $h_i \leftarrow \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)^\top \bar{\mu}_i$	-
Computation of step direction		
$C_i = \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} + B\, K_i^x & B\, K_i^u & -\mathbb{1}_{n_\mathrm{x}} \end{bmatrix}, \quad c_i = e_i + B\, \Delta\tilde{K}_i,$		
(Exact Hessian): $A_i = \nabla^2_{w_i} l(\bar{w}_i) + H_i$ and $a_i = \nabla_{w_i} l(\bar{w}_i) + \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} + K_i^{x\top} B^\top \\ K_i^{u\top} B^\top \end{bmatrix} \bar{\lambda}_i - \begin{bmatrix} \mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0} \end{bmatrix} \bar{\lambda}_{i-1} + h_i$		
(Gauss-Newton): $A_i = \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top \frac{\partial F(\bar{w}_i)}{\partial w_i}$ and $\nabla_{w_i} l(\bar{w}_i) = \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top F(\bar{w}_i)$		
Solve the linear KKT system (6.10) such that $\bar{w}_i^+ \leftarrow \bar{w}_i + \Delta w_i$ and $\bar{\lambda}_i^+ \leftarrow \bar{\lambda}_i + \Delta\lambda_i$		
Expansion procedure for $i = 0, \dots, N-1$ (backward sweep)		
$\tilde{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + K_i^w \Delta w_i$ -	$\tilde{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + \tilde{K}_i^w \Delta w_i$ -	$\tilde{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + \tilde{K}_i^w \Delta w_i$ $\bar{K}_i^{w+} \leftarrow \bar{K}_i^w + \Delta K_i^w$
(Exact Hessian):	**(Exact Hessian)**:	**(Exact Hessian)**:
$\bar{\mu}_i^+ \leftarrow -\frac{\partial G_i}{\partial K_i}^{-\top} \left(B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \Delta z_i \right)$	$\bar{\mu}_i^+ \leftarrow \bar{\mu}_i - M_i^{-\top} \left(\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \Delta z_i \right)$	$\bar{\mu}_i^+ \leftarrow \bar{\mu}_i - M_i^{-\top} \left(\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \Delta z_i \right)$
(Gauss-Newton):	**(Gauss-Newton)**:	**(Gauss-Newton)**:
-	$\bar{\mu}_i^+ \leftarrow \bar{\mu}_i - M_i^{-\top} \left(\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ \right)$	-

Table 6.2: Overview of the presented algorithms for (inexact) Newton based lifted collocation integrators.

Algorithm 6 Newton-type optimization step, using the adjoint-based inexact lifted collocation integrator within direct multiple shooting (LC-IN).

Input: Current values $\bar{z}_i = (\bar{x}_i, \bar{u}_i, \bar{K}_i)$, $(\bar{\lambda}_i, \bar{\mu}_i)$ and M_i for $i = 0, \dots, N-1$.
Output: Updated values $\bar{z}_i^+$ and $(\bar{\lambda}_i^+, \bar{\mu}_i^+)$ for $i = 0, \dots, N-1$.

`Condensing procedure`

for $i = 0, \dots, N-1$ **do in parallel** (forward sweep)

Compute the values $\Delta\tilde{K}_i$ and $\tilde{K}_i^w$ using Eq. (6.33):
$\Delta\tilde{K}_i \leftarrow -M_i^{-1} G(\bar{z}_i)$ and $\tilde{K}_i^w \leftarrow -M_i^{-1} \frac{\partial G_i}{\partial w_i}$.

In case of second-order sensitivities, using Eq. (6.34):
$H_i \leftarrow {z_i^w}^\top \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle \, z_i^w$.
$h_i \leftarrow {z_i^w}^\top \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \, \Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \tilde{K}_i^w \right)^\top \bar{\mu}_i$.

end for

`Computation of step direction`

Solve the linear system (6.10) based on the data C_i, A_i and c_i, a_i in (6.18) and (6.19) for $i = 0, \dots, N-1$, in order to obtain the step $(\Delta W, \Delta\Lambda)$.
$\bar{w}_i^+ \leftarrow \bar{w}_i + \Delta w_i$ and $\bar{\lambda}_i^+ \leftarrow \bar{\lambda}_i + \Delta\lambda_i$.

`Expansion procedure`

for $i = 0, \dots, N-1$ **do in parallel** (backward sweep)

The full solution can be obtained using Eq. (6.35):
$\bar{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + \tilde{K}_i^w \Delta w_i$.
$\bar{\mu}_i^+ \leftarrow \bar{\mu}_i - M_i^{-\top} \left(\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \, \Delta z_i \right)$.

end for

6.3.2 Local Convergence for Newton-type Methods

Let us briefly recall the local contraction result for Newton-type methods, which we will use throughout this chapter to study local convergence for inexact lifted collocation. To discuss the local convergence of the adjoint-based inexact lifting scheme, we will first write it in a more compact notation starting with the KKT equations

$$\mathcal{F}(Y) := \begin{bmatrix} \nabla_Z \mathcal{L}^{\mathrm{c}}(Z, \Lambda, \mu) \\ E\,Z \\ G(Z) \end{bmatrix} = 0, \tag{6.36}$$

where $Y := (Z, \Lambda, \mu)$ denotes the concatenated variables. Then, each Newton-type iteration from Eq. (6.32) can be written as

$$\Delta Y = -\tilde{J}(\bar{Y})^{-1} \mathcal{F}(\bar{Y}). \tag{6.37}$$

Given a guess, $\bar{Y}$, the Jacobian approximation from Eq. (6.32) is

$$\tilde{J}(\bar{Y}) := \begin{bmatrix} A_{\mathrm{c}}(\bar{Y}) & E^\top & \tilde{D}(\bar{Z})^\top \\ E & \mathbb{0} & \mathbb{0} \\ \tilde{D}(\bar{Z}) & \mathbb{0} & \mathbb{0} \end{bmatrix} \approx J(\bar{Y}) := \frac{\partial \mathcal{F}(\bar{Y})}{\partial Y}. \tag{6.38}$$

Because the system of equations in (6.36) denotes the KKT conditions [232] for the direct collocation NLP in Eq. (6.7), a solution $\mathcal{F}(Y^\star) = 0$ by definition also needs to be a KKT point $(Z^\star, \Lambda^\star, \mu^\star)$ for the original NLP.

The Newton-type optimization method in Algorithm 6 can now be rewritten as the compact iteration (6.37). The convergence of this scheme then follows the classical and well-known local contraction theorem from [44, 82, 94, 252], which has already been stated earlier in Theorem 1.26. It provides a simple means of assessing the stability of a solution point $Y^\star$ if and only if $\kappa^\star = \rho\left(\tilde{J}(Y^\star)^{-1} J(Y^\star) - \mathbb{1}\right) < 1$, and therefore provides a guarantee of the existence of a neighborhood where the Newton-type iteration converges linearly to $Y^\star$ with the asymptotic contraction rate $\kappa^\star$.

Remark 6.7 *The adjoint-based inexact lifting scheme converges locally to a solution of the direct collocation NLP if the assumptions of Theorem 1.26 and condition* (1.18) *are satisfied. As mentioned earlier, it is therefore possible to use a fixed Jacobian approximation* $\tilde{D}_i := [G_{w_i}, M_i]$ *over the different Newton-type iterations in Eq.* (6.32) *where, additionally,* $G_{w_i} \approx \frac{\partial G(\bar{z}_i)}{\partial w_i}$. *Theorem 1.26 still holds for this case. It results in the computational advantage that the factorization of* $\tilde{D}_i$ *needs to be computed only once. Additionally, the inexact forward sensitivities* $\tilde{K}_i^w = -M_i^{-1} G_{w_i}$ *remain fixed and can be computed offline. The use of fixed sensitivity approximations can also reduce the memory requirement for the lifted collocation integrator considerably [263].*

6.4 Inexact Newton with Iterated Sensitivities

This section proposes an alternative inexact lifted collocation integrator, based on an iterative scheme to compute the corresponding sensitivities. We formulate this technique as an inexact Newton method for an augmented KKT system and we briefly discuss its local convergence properties. Using the same principles of condensing and expansion, this inexact lifting scheme can be implemented similar to a direct multiple shooting based Newton-type optimization algorithm. Finally, we present the adjoint-free iterative inexact lifted collocation integrator as a special case of this approach.

6.4.1 Iterative Inexact Lifted Collocation Scheme

An inexact Newton scheme uses the factorization of one of the aforementioned approximations of the Jacobian $M_i \approx \frac{\partial G(\bar{w}_i, \bar{K}_i)}{\partial K_i}$ for each linear system solution. To recover the correct sensitivities in the Newton-type optimization algorithm, our proposed Inexact Newton scheme with Iterated Sensitivities (INIS) additionally includes lifting the forward sensitivities K_i^w. More specifically, the sensitivities K_i^w are introduced as extra variables into the NLP, which can be iteratively obtained by applying a Newton-type iteration to the linear equation $\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} K_i^w = 0$. Note that this idea is similar to the iterative differentiation technique in Section 2.3.6. The lifting of the sensitivities results in additional degrees of freedom such that the update for the collocation variables becomes $\Delta K_i = \Delta \tilde{K}_i + \bar{K}_i^w \Delta w_i$, where $\bar{K}_i^w$ denotes the current values for the lifted sensitivities. The resulting condensing procedure reads as

$$\begin{aligned} \Delta \tilde{K}_i &= -M_i^{-1} G(\bar{z}_i), \\ \Delta K_i^w &= -M_i^{-1} \left(\frac{\partial G(\bar{z}_i)}{\partial w_i} + \frac{\partial G(\bar{z}_i)}{\partial K_i} \bar{K}_i^w \right), \end{aligned} \tag{6.39}$$

instead of the standard inexact Newton step in Eq. (6.33).

In the case of a Newton-type optimization algorithm that requires the propagation of second-order sensitivities, one can apply a similar inexact update to the Lagrange multipliers μ_i corresponding to the collocation equations. The Newton-type scheme can equivalently be applied to the expression from Eq. (6.20), to result in the following iterative update

$$\Delta \mu_i = -M_i^{-\top} \left(\frac{\partial G(\bar{z}_i)}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \, \Delta z_i \right), \tag{6.40}$$

where $\bar{\mu}_i$ denotes the current values of the Lagrange multipliers corresponding to the collocation equations. The inexact Newton iteration (6.40) only requires the factorization of the Jacobian approximation M_i and corresponds to the expansion step in Eq. (6.35) for the adjoint-based inexact lifted collocation. The Newton-type optimization algorithm based on the inexact lifting scheme with *iterated sensitivities* (LC-INIS) within multiple shooting is detailed in Algorithm 7. A more general discussion on the INIS algorithm and its local convergence properties can be found in Chapter 7.

Iterative inexact lifting as an augmented Newton scheme

By introducing the (possibly fixed) Jacobian approximation $M_i \approx \frac{\partial G(\bar{w}_i, \bar{K}_i)}{\partial K_i}$ and the lifted variables for the forward sensitivities K_i^w for $i = 0, \ldots, N-1$, let

us define the following *augmented* and inexact version of the linearized KKT system (6.11) for direct collocation

$$\begin{bmatrix} A_{\mathrm{c}} & E^\top & \tilde{D}^\top & \mathbb{0} \\ E & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \tilde{D} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\mathrm{W}}} \otimes M \end{bmatrix} \begin{bmatrix} \Delta Z \\ \Delta\Lambda \\ \Delta\mu \\ \mathrm{vec}(\Delta K^w) \end{bmatrix} = - \begin{bmatrix} a_{\mathrm{c}} \\ e \\ d \\ d_{\mathrm{f}} \end{bmatrix}, \tag{6.41}$$

where the matrix $A_{\mathrm{c}} \in \mathbb{R}^{n_{\mathrm{Z}} \times n_{\mathrm{Z}}}$ is block diagonal and defined earlier in Eq. (6.11), and where $A_{\mathrm{c},i} := \nabla^2_{z_i} \mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i)$ and $\mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i) = l(\bar{w}_i) + \bar{\lambda}_i^\top \left(\bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}\right) + \bar{\mu}_i^\top G(\bar{w}_i, \bar{K}_i)$. Also, the constant matrix $E \in \mathbb{R}^{(N+1)n_{\mathrm{x}} \times n_{\mathrm{Z}}}$ is defined as before in Eq. (6.12). In addition, the block diagonal matrix $\tilde{D}$ is defined by $\tilde{D}_i = \left[-M_i \bar{K}_i^w,\, M_i\right] \in \mathbb{R}^{n_{\mathrm{K}} \times n_{\mathrm{z}}}$ for each $i = 0, \ldots, N-1$, because of the following

$$\begin{aligned} \tilde{D}_i &= \left[-M_i \bar{K}_i^w,\, M_i\right] \\ &\approx D_i = \left[-\frac{\partial G(\bar{w}_i, \bar{K}_i)}{\partial K_i} K_i^w,\, \frac{\partial G(\bar{w}_i, \bar{K}_i)}{\partial K_i}\right] = \frac{\partial G(\bar{z}_i)}{\partial z_i}, \end{aligned} \tag{6.42}$$

where the Jacobian approximation $M_i \approx \frac{\partial G(\bar{w}_i, \bar{K}_i)}{\partial K_i}$ is used. The following terms on the right-hand side are defined as before in Eq. (6.11) where $a_{\mathrm{c},i} := \nabla_{z_i} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu})$, $e_i := \bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}$ and $d_i := G(\bar{z}_i)$. In addition, the remaining terms read as $d_{\mathrm{f},i} := \mathrm{vec}(\frac{\partial G(\bar{z}_i)}{\partial w_i} + \frac{\partial G(\bar{z}_i)}{\partial K_i} \bar{K}_i^w)$. The following proposition states the connection between this augmented KKT system (6.41) and Algorithm 7 for an iterative inexact lifted collocation integrator.

Proposition 6.8 *Algorithm 7 presents a condensing technique for the augmented and inexact KKT system* (6.41)*, which allows one to instead solve the multiple shooting type system* (6.10)*. The original solution* $(\Delta Z, \Delta\Lambda, \Delta\mu, \Delta K^w)$ *can be obtained by use of the corresponding expansion step.*

Proof. Similar to the proof for Proposition 6.2, let us look closely at the first line of the KKT system in Eq. (6.41),

$$\nabla^2_{z_i} \mathcal{L}^{\mathrm{c}}_i \Delta z_i + E_i^\top \Delta\lambda_i - \begin{bmatrix} \mathbb{1}_{n_{\mathrm{x}}} \\ \mathbb{0} \\ \mathbb{0} \end{bmatrix} \Delta\lambda_{i-1} + \tilde{D}_i^\top \Delta\mu_i = -a_{\mathrm{c},i}. \tag{6.43}$$

For the inexact Newton case, we observe that the following holds

$$\tilde{D}_i z_i^w = -M_i \bar{K}_i^w + M_i \bar{K}_i^w = \mathbb{0},$$

using the approximate Jacobian matrices $z_i^{w\top} = \begin{bmatrix}\mathbb{1}_{n_\mathrm{w}} & \bar{K}_i^{w\top}\end{bmatrix}$ and $\tilde{D}_i = \begin{bmatrix}-M_i\bar{K}_i^w, M_i\end{bmatrix}$. We can multiply Eq. (6.43) on the left by $z_i^{w\top}$ and use $\Delta z_i = \begin{bmatrix}\mathbb{1}_{n_\mathrm{w}} \\ \bar{K}_i^w\end{bmatrix}\Delta w_i + \begin{bmatrix}\mathbb{0} \\ \mathbb{1}_{n_\mathrm{K}}\end{bmatrix}\Delta\tilde{K}_i$ to obtain the expression

$$\tilde{A}_i\Delta w_i + \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} + \bar{K}_i^{x\top}B^\top \\ \bar{K}_i^{u\top}B^\top\end{bmatrix}\Delta\lambda_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0}\end{bmatrix}\Delta\lambda_{i-1} = -\tilde{a}_i, \tag{6.44}$$

where the Hessian matrix $\tilde{A}_i = \nabla^2_{w_i} l(\bar{w}_i) + H_i$ with $H_i = z_i^{w\top}\langle\bar{\mu}_i, \nabla^2_{z_i} G_i\rangle\, z_i^w$. Furthermore, the right-hand side reads

$$\begin{aligned}\tilde{a}_i &= z_i^{w\top}\nabla_{z_i}\mathcal{L}^\mathrm{c} + z_i^{w\top}\nabla^2_{z_i,K_i}\mathcal{L}_i^\mathrm{c}\Delta\tilde{K}_i \\ &= \nabla_{w_i} l(\bar{w}_i) + \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} + \bar{K}_i^{x\top}B^\top \\ \bar{K}_i^{u\top}B^\top\end{bmatrix}\bar{\lambda}_i - \begin{bmatrix}\mathbb{1}_{n_\mathrm{x}} \\ \mathbb{0}\end{bmatrix}\bar{\lambda}_{i-1} + \tilde{h}_i,\end{aligned}$$

where $\tilde{h}_i = z_i^{w\top}\langle\bar{\mu}_i, \nabla^2_{z_i,K_i} G_i\rangle\,\Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i}\bar{K}_i^w\right)^\top\bar{\mu}_i$. The augmented KKT system (6.41) can therefore indeed be reduced to the multiple shooting type form in Eq. (6.10), using the condensing step as described in Algorithm 7.

The expansion step for the lifted K variables follows from $\tilde{D}_i\Delta z_i = -d_i$ and becomes $\Delta K_i = \Delta\tilde{K}_i + \bar{K}_i^w\Delta w_i$. To update the Lagrange multipliers μ_i, let us look at the lower part of Eq. (6.43):

$$\nabla^2_{K_i,z_i}\mathcal{L}_i^\mathrm{c}\Delta z_i + B^\top\Delta\lambda_i + M_i^\top\Delta\mu_i = -\nabla_{K_i}\mathcal{L}^\mathrm{c},$$

which can be rewritten as $\Delta\mu_i = -M_i^{-\top}\left(\frac{\partial G(\bar{z}_i)}{\partial K_i}^\top\bar{\mu}_i + B^\top\bar{\lambda}_i^+ + \langle\bar{\mu}_i, \nabla^2_{K_i,z_i} G_i\rangle\,\Delta z_i\right)$ in Equation (6.40). Finally, the update of the lifted sensitivities K_i^w follows from the last equation of the KKT system in (6.41)

$$\Delta K_i^w = -M_i^{-1}\left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i}\bar{K}_i^w\right).$$

□

Remark 6.9 *To be precise, Algorithm 7 is an adjoint-based iterative inexact lifting scheme since it corrects the gradient in the condensed problem* (6.44) *using the expression* $\left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i}\bar{K}_i^w\right)^\top\bar{\mu}_i$ *similar to Eq.* (6.34) *for the adjoint-based inexact scheme. This correction term is equal to zero whenever the lifted sensitivities are exact, i.e.,* $K_i^{w\star} = -\frac{\partial G_i}{\partial K_i}^{-1}\frac{\partial G_i}{\partial w_i}$. *The overview in Table 6.2 allows one to compare this novel approach for inexact Newton based lifted collocation with the previously presented lifting schemes.*

Algorithm 7 Newton-type optimization step, based on the iterative inexact lifted collocation integrator within direct multiple shooting (LC-INIS).

Input: Current values $\bar{z}_i = (\bar{x}_i, \bar{u}_i, \bar{K}_i)$, $\bar{K}_i^w$, $(\bar{\lambda}_i, \bar{\mu}_i)$ and M_i, $i = 0, \dots, N-1$.
Output: Updated values $\bar{z}_i^+$, $\bar{K}_i^{w^+}$ and $(\bar{\lambda}_i^+, \bar{\mu}_i^+)$ for $i = 0, \dots, N-1$.

`Condensing procedure`

1: **for** $i = 0, \dots, N-1$ **do in parallel** (forward sweep)

2: Compute the values $\Delta\tilde{K}_i$ and ΔK_i^w using Eq. (6.39):
$\Delta\tilde{K}_i \leftarrow -M_i^{-1} G(\bar{z}_i)$ and $\Delta K_i^w \leftarrow -M_i^{-1} \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \bar{K}_i^w \right)$.

3: In case of second-order sensitivities, using Eq. (6.44):
$H_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i} G_i \rangle \, z_i^w$.
$h_i \leftarrow z_i^{w\top} \langle \bar{\mu}_i, \nabla^2_{z_i, K_i} G_i \rangle \, \Delta\tilde{K}_i + \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \bar{K}_i^w \right)^\top \bar{\mu}_i$.

4: **end for**

`Computation of step direction`

5: Solve the linear system (6.10) based on the data C_i, A_i and c_i, a_i in (6.18) and (6.19) for $i = 0, \dots, N-1$, in order to obtain the step $(\Delta W, \Delta\Lambda)$.
$\bar{w}_i^+ \leftarrow \bar{w}_i + \Delta w_i$ and $\bar{\lambda}_i^+ \leftarrow \bar{\lambda}_i + \Delta\lambda_i$.

`Expansion procedure`

6: **for** $i = 0, \dots, N-1$ **do in parallel** (backward sweep)

7: The full solution can be obtained using Eq. (6.40):
$\bar{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + \bar{K}_i^w \Delta w_i$ and $\bar{K}_i^{w^+} \leftarrow \bar{K}_i^w + \Delta K_i^w$.
$\bar{\mu}_i^+ \leftarrow \bar{\mu}_i - M_i^{-\top} \left(\frac{\partial G_i}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \langle \bar{\mu}_i, \nabla^2_{K_i, z_i} G_i \rangle \, \Delta z_i \right)$.

8: **end for**

Remark 6.10 *The updates of the lifted forward sensitivities,*

$$\Delta K_i^w = -M_i^{-1} \left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i} \bar{K}_i^w \right), \tag{6.45}$$

are independent of the updates for any of the other primal or dual variables, so that (6.45) *can be implemented separately. More specifically, one can carry out multiple Newton-type iterations for the lifted variables $\bar{K}_i^w$ followed by an update of the remaining variables or the other way around. To simplify our discussion on the local convergence for this INIS type optimization algorithm, we will however not consider such variations further.*

6.4.2 Local Convergence for Iterative Inexact Lifting

Similar to Section 6.3.2, we introduce a more compact notation for the Newton-type iteration from Algorithm 7. For this purpose, we define the following augmented system of KKT equations:

$$\mathcal{F}_{\text{INIS}}(Y_{\text{a}}) := \begin{bmatrix} \nabla_Z \mathcal{L}^{\text{c}}(Z, \Lambda, \mu) \\ E\,Z \\ G(Z) \\ \text{vec}(\frac{\partial G(Z)}{\partial W} + \frac{\partial G(Z)}{\partial K} K^w) \end{bmatrix} = 0, \tag{6.46}$$

where the concatenated variables $Y_{\text{a}} := (Z, \Lambda, \mu, K^w)$ are defined. The INIS type iteration then reads as $\tilde{J}_{\text{INIS}}(\bar{Y}_{\text{a}})\Delta Y_{\text{a}} = -\mathcal{F}_{\text{INIS}}(\bar{Y}_{\text{a}})$ and uses the following Jacobian approximation

$$\begin{aligned} \tilde{J}_{\text{INIS}}(\bar{Y}_{\text{a}}) &:= \begin{bmatrix} A_{\text{c}}(\bar{Y}) & E^\top & \tilde{D}(\bar{Z}, \bar{K}^w)^\top & \mathbb{0} \\ E & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \tilde{D}(\bar{Z}, \bar{K}^w) & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{W}}} \otimes M(\bar{Z}) \end{bmatrix} \\ &\approx J_{\text{INIS}}(\bar{Y}_{\text{a}}) := \frac{\partial \mathcal{F}_{\text{INIS}}(\bar{Y}_{\text{a}})}{\partial Y_{\text{a}}}, \end{aligned} \tag{6.47}$$

where $Y := (Z, \Lambda, \mu)$ and using the Jacobian approximations $M_i(\bar{z}_i) \approx \frac{\partial G(\bar{z}_i)}{\partial K_i}$. We show next that a solution to the augmented system $\mathcal{F}_{\text{INIS}}(Y_{\text{a}}) = 0$ also forms a solution to the direct collocation NLP in Eq. (6.7).

Proposition 6.11 *A solution $Y_{\text{a}}^\star = (Z^\star, \Lambda^\star, \mu^\star, K^{w^\star})$ which satisfies the LICQ and SOSC conditions [232] for the nonlinear system $\mathcal{F}_{\text{INIS}}(Y_{\text{a}}) = 0$, forms a regular KKT point $(Z^\star, \Lambda^\star, \mu^\star)$ for the direct collocation NLP in Eq.* (6.7)*.*

Proof. The proof follows directly from observing that the first three equations of the augmented system (6.46) correspond to the KKT conditions for the direct collocation NLP in Eq. (6.7). A solution $Y_{\text{a}}^\star$ of the system $\mathcal{F}_{\text{INIS}}(Y_{\text{a}}) = 0$ then provides a regular KKT point $(Z^\star, \Lambda^\star, \mu^\star)$ for this NLP (6.7). □

The Newton-type optimization method from Algorithm 7 has been rewritten as the compact iteration $\tilde{J}_{\text{INIS}}(\bar{Y}_{\text{a}})\Delta Y_{\text{a}} = -\mathcal{F}_{\text{INIS}}(\bar{Y}_{\text{a}})$. The local convergence properties of this scheme are described by the classical Newton-type contraction theory [44]. Under the conditions from Theorem 1.26, the iterates converge linearly to the solution $Y_{\text{a}}^\star$ with the asymptotic contraction rate

$$\kappa_{\text{INIS}}^\star = \rho\left(\tilde{J}_{\text{INIS}}(Y_{\text{a}}^\star)^{-1} J_{\text{INIS}}(Y_{\text{a}}^\star) - \mathbb{1}\right) < 1. \tag{6.48}$$

A more detailed discussion on this local contraction rate for an INIS type optimization algorithm and a comparison to standard adjoint-based inexact Newton schemes can be found in Chapter 7.

6.4.3 Adjoint-free Iterative Inexact Lifted Collocation

The inexact Newton scheme with iterated sensitivities from Algorithm 7 is based on an adjoint propagation to compute the correct gradient of the Lagrangian on the right-hand side of the KKT system from Eq. (6.41). Because of the lifting of the forward sensitivities K_i^w for $i = 0, \dots, N-1$, one can however avoid the computation of such an adjoint and still obtain a Newton-type algorithm that converges to a solution of the direct collocation NLP (6.7).

For this purpose, let us introduce the following *adjoint-free* approximation of the augmented KKT system in Eq. (6.41),

$$\begin{bmatrix} A_{\mathrm{c}} & E^\top & \tilde{D}^\top & \mathbb{0} \\ E & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \tilde{D} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\mathrm{W}}} \otimes M \end{bmatrix} \begin{bmatrix} \Delta Z \\ \Delta\Lambda \\ \Delta\mu \\ \mathrm{vec}(\Delta K^w) \end{bmatrix} = - \begin{bmatrix} \tilde{a}_{\mathrm{c}} \\ e \\ d \\ d_{\mathrm{f}}, \end{bmatrix}, \tag{6.49}$$

where all quantities are defined as in Eq. (6.41) but an approximate Lagrangian gradient term is used, i.e.,

$$\tilde{a}_{\mathrm{c},i} := \nabla_{z_i} l(\bar{w}_i) + \begin{bmatrix} \bar{\lambda}_i - \bar{\lambda}_{i-1} \\ \mathbb{0} \\ B^\top \bar{\lambda}_i \end{bmatrix} + \hat{D}_i^\top \bar{\mu}_i \approx \nabla_{z_i} \mathcal{L}^{\mathrm{c}}(\bar{Z}, \bar{\Lambda}, \bar{\mu}), \tag{6.50}$$

where $\hat{D}_i = \left[-\frac{\partial G(\bar{z}_i)}{\partial K_i} \bar{K}_i^w, \frac{\partial G(\bar{z}_i)}{\partial K_i} \right] \approx \frac{\partial G(\bar{z}_i)}{\partial z_i}$. Proposition 6.8 then still holds for this variant of lifted collocation, where the multiple shooting type gradient is instead defined without the adjoint-based correction term. The resulting algorithm is therefore referred to as an *adjoint-free* scheme (LC-AF-INIS) and it is detailed further in Algorithm 8 based on a Gauss-Newton Hessian approximation. It is important for the study of its local convergence that $\hat{D}_i \neq \tilde{D}_i = \left[-M_i \bar{K}_i^w, M_i \right]$, where $\tilde{D}_i$ is used in the Jacobian approximation and $\hat{D}_i$ is merely used to define the augmented KKT system in Eq. (6.49).

Local convergence for adjoint-free INIS scheme (AF-INIS)

To study the local convergence properties for the adjoint-free variant of the INIS based lifted collocation scheme, we need to investigate the approximate

Algorithm 8 Adjoint-free and multiplier-free Newton-type optimization step, based on Gauss-Newton and the iterative inexact lifted collocation integrator within direct multiple shooting (LC-AF-INIS).

Input: Current values $\bar{z}_i = (\bar{x}_i, \bar{u}_i, \bar{K}_i)$, $\bar{K}_i^w$ and M_i for $i = 0, \dots, N-1$.
Output: Updated values $\bar{z}_i^+$ and $\bar{K}_i^{w^+}$ for $i = 0, \dots, N-1$.

`Condensing procedure`

1: **for** $i = 0, \dots, N-1$ **do in parallel**

2: Compute the values $\Delta\tilde{K}_i$ and ΔK_i^w using Eq. (6.39):
$\Delta\tilde{K}_i \leftarrow -M_i^{-1} G(\bar{z}_i)$ and $\Delta K_i^w \leftarrow -M_i^{-1}\left(\frac{\partial G_i}{\partial w_i} + \frac{\partial G_i}{\partial K_i}\bar{K}_i^w\right)$.

3: $H_i \leftarrow \mathbb{0}$ and $h_i \leftarrow \mathbb{0}$.

4: **end for**

`Computation of step direction`

5: Solve the linear system (6.10) based on the data C_i, A_i and c_i, a_i in (6.18) and (6.19), in order to obtain the step ΔW and perform the primal update $\bar{w}_i^+ \leftarrow \bar{w}_i + \Delta w_i$ for $i = 0, \dots, N-1$.

$$A_i \leftarrow \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top \frac{\partial F(\bar{w}_i)}{\partial w_i} \text{ and } \nabla_{w_i} l(\bar{w}_i) \leftarrow \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top F(\bar{w}_i). \qquad \text{(Gauss-Newton)}$$

`Expansion procedure`

6: **for** $i = 0, \dots, N-1$ **do in parallel**

7: The full solution can be obtained:
$\bar{K}_i^+ \leftarrow \bar{K}_i + \Delta\tilde{K}_i + \bar{K}_i^w \Delta w_i$ and $\bar{K}_i^{w^+} \leftarrow \bar{K}_i^w + \Delta K_i^w$.

8: **end for**

augmented KKT system from Eq. (6.49). It is written as $\tilde{J}_{\text{INIS}}(\bar{Y}_\text{a})\Delta Y_\text{a} = -\mathcal{F}_{\text{AF}}(\bar{Y}_\text{a})$ in its compact form, where $\mathcal{F}_{\text{AF}}(Y_\text{a}) = 0$ represents the following approximate augmented system of KKT equations,

$$\mathcal{F}_{\text{AF}}(Y_\text{a}) := \begin{bmatrix} \nabla_Z \tilde{\mathcal{L}}^\text{c}(Z, \Lambda) + \hat{D}(Z, K^w)^\top \mu \\ E\,Z \\ G(Z) \\ \text{vec}(\frac{\partial G(Z)}{\partial W} + \frac{\partial G(Z)}{\partial K} K^w) \end{bmatrix} = 0, \tag{6.51}$$

where the incomplete Lagrangian $\tilde{\mathcal{L}}_i^\text{c}(\bar{z}_i, \bar{\lambda}_i) = l(\bar{w}_i) + \bar{\lambda}_i^\top \left(\bar{x}_i + B\,\bar{K}_i - \bar{x}_{i+1}\right)$ and the approximate Jacobian $\hat{D}_i = \left[-\frac{\partial G(\bar{z}_i)}{\partial K_i}\bar{K}_i^w,\ \frac{\partial G(\bar{z}_i)}{\partial K_i}\right]$ are defined. Note that the Jacobian approximation $\tilde{J}_{\text{INIS}}(\bar{Y}_\text{a})$ in the Newton-type iteration is still defined by Eq. (6.47), equivalent to the adjoint-based INIS scheme. The following proposition then shows that a solution to the approximate augmented system $\mathcal{F}_{\text{AF}}(Y_\text{a}) = 0$ is also a local minimizer for the direct collocation NLP (6.7).

Proposition 6.12 *A solution* $Y_{\mathrm{a}}^\star = (Z^\star, \Lambda^\star, \mu^\star, K^{w^\star})$ *which satisfies the LICQ and SOSC conditions [232] for the system of nonlinear equations* $\mathcal{F}_{\mathrm{AF}}(Y_{\mathrm{a}}) = 0$*, also forms a solution to the nonlinear system* $\mathcal{F}_{\mathrm{INIS}}(Y_{\mathrm{a}}) = 0$ *and therefore forms a regular KKT point* $(Z^\star, \Lambda^\star, \mu^\star)$ *for the direct collocation NLP in Eq.* (6.7)*.*

Proof. We observe that the lower part of the KKT system in Eq. (6.51) for the solution point $Y_{\mathrm{a}}^\star$ reads as

$$\frac{\partial G(z_i^\star)}{\partial w_i} + \frac{\partial G(z_i^\star)}{\partial K_i} K_i^{w^\star} = 0, \quad \text{for} \quad i = 0, \dots, N-1, \tag{6.52}$$

so that $K_i^{w^\star} = -\frac{\partial G(z_i^\star)}{\partial K_i}^{-1} \frac{\partial G(z_i^\star)}{\partial w_i}$ holds at any solution of $\mathcal{F}_{\mathrm{AF}}(Y_{\mathrm{a}}) = 0$. The same holds at a solution of $\mathcal{F}_{\mathrm{INIS}}(Y_{\mathrm{a}}) = 0$ in (6.46). Subsequently, we observe that $\hat{D}_i(z_i^\star, K_i^{w^\star}) = \left[-\frac{\partial G(z_i^\star)}{\partial K_i} K_i^{w^\star},\ \frac{\partial G(z_i^\star)}{\partial K_i}\right] = \frac{\partial G(z_i^\star)}{\partial z_i}$ such that the following equality holds at the solution

$$\nabla_Z \tilde{\mathcal{L}}^{\mathrm{c}}(Z^\star, \Lambda^\star) + \hat{D}(Z^\star, K^{w^\star})^\top \mu^\star = \nabla_Z \mathcal{L}^{\mathrm{c}}(Z^\star, \Lambda^\star, \mu^\star).$$

It follows that $Y_{\mathrm{a}}^\star$ forms a solution to the original augmented KKT system from Eq. (6.46). The result then follows directly from Proposition 6.11. □

Under the conditions of Theorem 1.26, the iterates defined by the Newton-type iteration $\tilde{J}_{\mathrm{INIS}}(\bar{Y}_{\mathrm{a}})\Delta Y_{\mathrm{a}} = -\mathcal{F}_{\mathrm{AF}}(\bar{Y}_{\mathrm{a}})$ converge linearly to the solution $Y_{\mathrm{a}}^\star$ with the asymptotic contraction rate

$$\kappa_{\mathrm{AF}}^\star = \rho\left(\tilde{J}_{\mathrm{INIS}}(Y_{\mathrm{a}}^\star)^{-1} J_{\mathrm{AF}}(Y_{\mathrm{a}}^\star) - \mathbb{1}\right) < 1, \tag{6.53}$$

based on the approximation $\tilde{J}_{\mathrm{INIS}}(Y_{\mathrm{a}}) \approx J_{\mathrm{AF}}(Y_{\mathrm{a}}) := \frac{\partial \mathcal{F}_{\mathrm{AF}}(Y_{\mathrm{a}})}{\partial Y_{\mathrm{a}}}$ from (6.47).

Adjoint-free and multiplier-free INIS based on Gauss-Newton

The motivation for the alternative INIS-type lifting scheme proposed in the previous subsection is to avoid the computation of any adjoint derivatives, while maintaining a Newton-type optimization algorithm that converges to a local minimizer of the direct collocation NLP. This equivalence result has been established in Proposition 6.12. However, the propagation of second-order sensitivities would still require the iterative update of the Lagrange multipliers, $\Delta\mu_i = -M_i^{-\top}\left(\frac{\partial G(\bar{z}_i)}{\partial K_i}^\top \bar{\mu}_i + B^\top \bar{\lambda}_i^+ + \nabla^2_{K_i, z_i} \mathcal{L}_i^{\mathrm{c}}\, \Delta z_i\right)$, based on adjoint differentiation. This alternative implementation would therefore not result in a considerable advantage over the standard INIS method.

Instead, the benefits for this adjoint-free scheme are more clear in case of a least squares objective $l(w_i) = \frac{1}{2}\|F(w_i)\|_2^2$ where one can use a Gauss-Newton (GN) approximation $A_{\mathrm{c},i} := \begin{bmatrix} \frac{\partial F(\bar{w}_i)}{\partial w_i}^\top \frac{\partial F(\bar{w}_i)}{\partial w_i} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} \end{bmatrix} \approx \nabla^2_{z_i} \mathcal{L}^{\mathrm{c}}_i(\bar{z}_i, \bar{\lambda}_i, \bar{\mu}_i)$ for the Hessian of the Lagrangian [48]. In that case, the Jacobian approximation for the augmented KKT system (6.47) is independent of the Lagrange multipliers as discussed earlier in Section 6.2.4. After applying Proposition 6.8 to condense this approximate augmented KKT system of the form of Eq. (6.49) to the multiple shooting type system in (6.10), the resulting scheme therefore does not depend on any of the Lagrange multipliers. This *adjoint-free* and *multiplier-free* implementation of Gauss-Newton based INIS type lifted collocation within multiple shooting is detailed in Algorithm 8.

6.5 Lifted Collocation in ACADO Code Generation

Let us provide a brief overview of the different proposed schemes for lifted collocation, including a discussion on their relative advantages and disadvantages. Table 6.3 presents a classification of all the variants of lifted collocation integrators which have been presented in this chapter. Note that these variants of collocation based optimal control algorithms from Table 6.3 are additionally implemented in the open-source `ACADO Toolkit` [176] software. The collocation methods are based on either Gauss-Legendre or Radau IIA points [158] and the proposed Jacobian approximations are based on either Simplified or Single Newton-type iterations as discussed in Section 2.4. The software is free of charge and can be downloaded from `www.acadotoolkit.org`, but more information can also be found in Chapter 8.

The most distinguishing characteristic in the overview of Table 6.3 is whether the algorithm is based on exact (LC-EN) or inexact lifting, discussed respectively in Section 6.2 and in Sections 6.3-6.4. Unlike the inexact lifting techniques, exact lifted collocation relies on computing a factorization of the Jacobian of the collocation equations. However, one can still choose either an exact Hessian or a Gauss-Newton type approximation within the optimization algorithm as shown in Table 6.3. Among the inexact lifting schemes, we make a distinction between the standard adjoint-based technique (LC-IN) from Section 6.3 and the inexact Newton scheme with iterated sensitivities (INIS) from Section 6.4. The latter INIS-type algorithm allows for an adjoint-based (LC-INIS) as well as an adjoint-free and multiplier-free (LC-AF-INIS) implementation using Gauss-Newton. Table 6.3 additionally includes multiple shooting (MS) without lifting the collocation variables and direct collocation (DC). For the standard (MS)

implementation, a method to solve the nonlinear system of collocation equations needs to be embedded within the Newton-type optimization algorithm [263, 271]. Similar to (DC), all lifted collocation type schemes avoid such internal iterations as mentioned earlier in Table 6.1.

The main advantage of the inexact schemes (LC-IN) and (LC-INIS) over the exact lifted collocation (LC-EN) is the reduced computational effort. Even though their local convergence is generally slower (see Theorem 1.26), the cost per iteration can be reduced considerably based on the use of a Jacobian approximation for the system of collocation equations. Since a relatively low accuracy of the solution is often sufficient, e.g., for real-time optimal control on embedded applications [90, 315], the overall computational cost can be much better for inexact Newton-based lifting. Between the two families of inexact schemes, the INIS algorithm results typically in better local contraction properties as illustrated by the numerical case study in the next section. In addition, it allows for an adjoint-free implementation in Scheme (LC-AF-INIS) for optimal control problems involving a least squares type objective as described by Algorithm 8. A detailed discussion on the local convergence for the INIS based optimization algorithms can be found in Chapter 7.

Regarding memory requirements for the various lifting schemes in Table 6.3, it is important to note that any algorithm based on an adjoint sweep requires the storage of variables during the preceding forward sweep as discussed in Section 6.2.3. This is a benefit for the GN based exact lifting (LC-EN) and adjoint-free INIS scheme (LC-AF-INIS), because both rely only on forward propagation of sensitivities. Another noticeable effect is the storage of the first-order forward sensitivities K^w in both exact and INIS-type lifting. The adjoint-based inexact lifting (LC-IN) has the advantage that one could use Jacobian approximations, which are precomputed and fixed in order to further reduce the memory requirements of the corresponding implementation at the cost of possibly slowing down the local convergence. In the case of an INIS type algorithm, these forward sensitivities are additionally lifted and therefore need to be stored from one iteration to the next.

6.6 Case Study: Chain of Masses

This section illustrates the performance of the proposed variants of lifted implicit integrators on the benchmark optimal control example, which consists of a chain of spring connected masses. For this purpose, a multiple shooting based SQP method is implemented using the `ACADO` code generation tool. In the numerical results of this chapter, the QP solutions are obtained using the active-set

Table 6.3: Overview on the different variants of collocation based optimal control algorithms (EH = Exact Hessian, GN = Gauss-Newton).

Scheme	Algorithm	Newton type	Hessian type
(LC-EN)	Alg. 5	exact lifting	GN or EH
(LC-IN)	Alg. 6	adjoint-based	GN or EH
(LC-INIS)	Alg. 7	adjoint-based INIS	GN or EH
(LC-AF-INIS)	Alg. 8	adjoint-free INIS	GN
(MS)	Eq. (6.10)	without lifting	GN or EH
(DC)	Eq. (6.11)	fully sparse	GN or EH

solver `qpOASES` [109] in combination with a condensing technique to numerically eliminate the state variables [51]. Mainly as a reference, the direct collocation problem is additionally solved using the general-purpose sparse NLP solver `Ipopt` [318] based on the interface in the `CasADi` software [20]. Note however that these two implementations cannot be compared directly, since `Ipopt` is a general-purpose solver and therefore includes many additional features. On the other hand, the `ACADO` generated SQP method can be warm started and respects all linearized constraints at each iteration, which are both important features for real-time applications of optimal control [90]. We will therefore additionally report the computation times for solving the direct collocation QP subproblem, using a general-purpose sparse solver in the `OOQP` software [132]. Note that both the numerical results with `OOQP` and with `Ipopt` are based on the `MA27` linear algebra code from the HSL library [4], in order to solve the sparse linear system at each interior point iteration.

All numerical simulations are carried out on a standard computer, equipped with Intel i7-3720QM processor, using a 64-bit version of Ubuntu 14.04 and the g++ compiler version 4.8.4. The presented results can be verified by running the `MATLAB` simulation scripts that can be found on the following public repository: `https://github.com/rienq/liftedCollocation`.

6.6.1 Optimal Control Problem Formulation

We consider the chain mass control problem, which was used to illustrate the compression algorithm for distributed multiple shooting in Chapter 5. The task of the controller is to return a chain of n_m masses connected with springs to its steady state, starting from a perturbed initial configuration. The mass at one end is fixed, while the control input $u \in \mathbb{R}^3$ to the system is the direct force applied to the mass at the other end of the chain. The OCP formulation

includes simple bounds on the control inputs and the state constraint that the chain should not hit a wall placed close to the equilibrium state as illustrated by Figure 5.1, i.e., $p_y^j \geq -0.01$ for $j = 1, \ldots, n_\mathrm{m} - 1$ where p_y^j denotes the position of the free mass in direction of the y-axis. In addition, both the initial and terminal state are constrained resulting in

$$\begin{aligned}
\min_{x(\cdot),\, u(\cdot)} \quad & \int_0^T \ell(x(t), u(t))\, \mathrm{d}t && && \text{(6.54a)}\\
\text{s.t.} \quad 0 &= x(0) - \hat{x}_0, && && \text{(6.54b)}\\
\dot{x}(t) &= f_\mathrm{chain}(x(t), u(t)), && \forall t \in [0, T], && \text{(6.54c)}\\
0 &= x(T) - \hat{x}_T, && && \text{(6.54d)}\\
-10 &\leq u(t) \leq 10, && \forall t \in [0, T], && \text{(6.54e)}\\
-0.01 &\leq p_y^j(t), \quad j = 1, \ldots, n_\mathrm{m} - 1, && \forall t \in [0, T], && \text{(6.54f)}
\end{aligned}$$

where $\hat{x}_0$ and $\hat{x}_T$ denote respectively the initial perturbed and the terminal equilibrium state values. The stage cost in the objective represents minimizing the control effort, such that $\ell_\mathrm{ME}(\cdot) := \|u(t)\|_2^2$ is defined. Note that such a least squares type objective will allow us to validate the Gauss-Newton based algorithms for this minimum-effort (ME) type OCP. Alternatively, we include a time optimal reformulation where we introduce the additional state variable T_opt such that the scaled dynamics read as

$$\begin{aligned}
\dot{x}(t) &= T_\mathrm{opt}\ f_\mathrm{chain}(x(t), u(t))\\
\dot{T}_\mathrm{opt}(t) &= 0,
\end{aligned} \tag{6.55}$$

which then replaces the original ODE model in Eq. (6.54c). The time scaling variable itself is not constrained, but instead forms the optimization objective $\ell_\mathrm{TO}(\cdot) := T_\mathrm{opt}$ in the time optimal (TO) formulation.

The horizon length is chosen to be $T = 5$s and a multiple shooting method is applied to the OCP (6.54), using $N = 20$ equidistant intervals. This results in a shooting interval of size $T_s = 0.25$s for the minimum-effort problem. For the time optimal formulation, the horizon length is instead taken $T = 1$s such that the scaling variable T_opt directly represents the time in which the point-to-point motion is carried out. Note that the definition of this additional state variable T_opt, allows us to preserve the block banded structure in the discrete time OCP (6.4). In both cases, three integration steps $N_\mathrm{s} = 3$ of a Gauss-Legendre collocation method using $q = 4$ stages, i.e., of order 8, are used within each shooting interval. The resulting nonlinear OCP will be solved for different

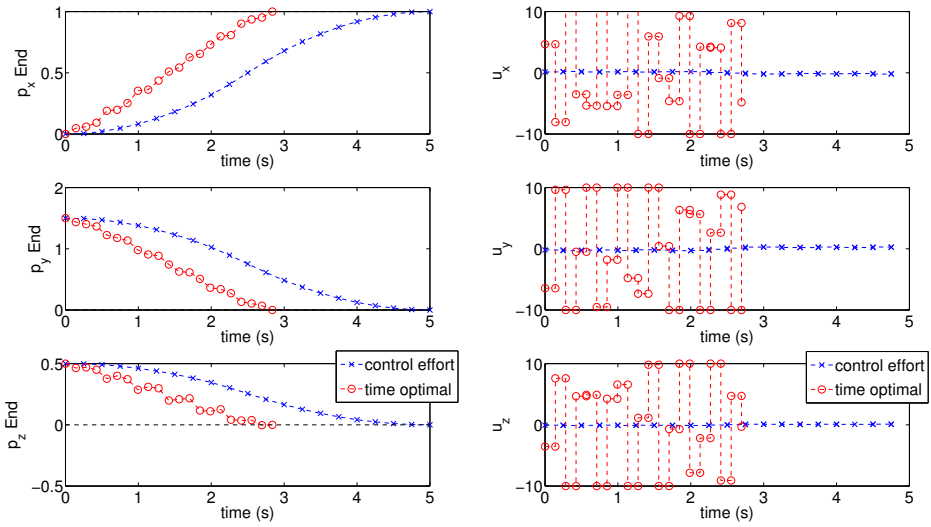


Figure 6.4: Minimizing the control effort versus time optimal OCP formulation for the chain mass example: optimal state and control trajectories ($n_{\mathrm{m}} = 8$).

numbers of masses n_{m} to illustrate the numerical performance of the lifted collocation integrators from Table 6.3. Figure 6.4 additionally shows the solution trajectories of the minimum-effort versus time optimal OCP formulation for $n_{\mathrm{m}} = 8$ masses, including the position $p^{n_{\mathrm{m}}-1}$ of the free mass at the controlled end of the chain.

6.6.2 Numerical Simulation Results

Table 6.4 shows the average computation times for the Gauss-Newton type SQP method on the minimum-effort OCP problem formulation. The table shows the average computation time per SQP iteration and this for different numbers of masses $n_{\mathrm{m}} = 3, \ldots, 7$. It includes the standard multiple shooting method (MS) without lifting, as well as exact lifted collocation (LC-EN) and the inexact lifting schemes (LC-IN) and (LC-AF-INIS). The table illustrates that for systems with more states, the computational benefit of using inexact Newton based lifting schemes can be considerably higher. Note that the Jacobian approximation in the (LC-IN) and the (LC-AF-INIS) schemes is based on the Single Newton-type iteration in these experiments. For a specific instance of

Table 6.4: Average Gauss-Newton based SQP timing results for the minimum effort chain mass OCP using 4-stage Gauss collocation ($N_\mathrm{s} = 3, q = 4$), including different numbers of masses n_m and resulting numbers of states n_x.

n_m	n_x	Without lifting (MS)	Exact lifting (LC-EN)	IN lifting (LC-IN)	INIS lifting (LC-AF-INIS)
3	12	17.63 ms	6.12 ms	1.93 ms	2.35 ms
4	18	40.46 ms	17.55 ms	4.48 ms	5.63 ms
5	24	73.37 ms	33.98 ms	8.29 ms	7.66 ms
6	30	145.58 ms	64.68 ms	13.61 ms	16.50 ms
7	36	242.41 ms	133.14 ms	22.92 ms	20.41 ms

the chain mass problem where $n_\mathrm{m} = 5$, more detailed timing results are shown in Table 6.5. It includes the average computation time spent in each component of the algorithm per SQP iteration, including the simulation with sensitivity propagation, condensing of the structured QP subproblem and the solution of the resulting condensed problem using `qpOASES`. It is the simulation time that can be reduced considerably by using lifted collocation integrators, which appears to account for the highest portion of the total computational effort for this numerical case study. More specifically, a speedup factor of about 2 can be observed when using lifted collocation instead of the standard method without lifting. When using the INIS-type lifting scheme, this computational speedup increases to a factor of about 10. Note that one iteration of the general-purpose sparse NLP solver `Ipopt` takes about 500 ms in this case, while the solution of one direct collocation based QP takes about 2.4 s using the sparse `OOQP` solver.

Table 6.6 shows the average computation times for an exact Hessian based SQP iteration on the time optimal OCP using different numbers of masses n_m while Table 6.7 presents the detailed timing results using $n_\mathrm{m} = 5$ masses. In a similar way as for the Gauss-Newton based implementation, it can be observed from the latter table that both the exact and inexact lifting schemes reduce the computational effort over the standard multiple shooting method. More specifically, a speedup factor of almost 2 can be observed when using the (LC-EN) scheme instead of the standard collocation integrator without lifting. The table additionally shows that the inexact lifted collocation integrators (LC-IN) and (LC-INIS) reduce the computation time less in the context of second-order sensitivity propagation, compared to the Gauss-Newton based implementation in Table 6.4 and 6.5. However, a computational speedup factor of about 5 can still be observed in Table 6.7 when using, for example, the INIS-type lifting

Table 6.5: Detailed timing results for Gauss-Newton based SQP on the minimum effort OCP using $n_{\mathrm{m}} = 5$ masses or $n_{\mathrm{x}} = 24$ states ($N_{\mathrm{s}} = 3, q = 4$). Note that one iteration of direct collocation (6.7) based on `Ipopt` takes about 500 ms, and one sparse QP solution using `OOQP` takes 2.4 s on average.

	Without lifting (MS)	Exact lifting (LC-EN)	IN lifting (LC-IN)	INIS lifting (LC-AF-INIS)
simulation	**71.86 ms**	32.48 ms	6.73 ms	**6.09 ms**
condensing	0.85 ms	0.84 ms	0.92 ms	0.90 ms
QP solution	0.60 ms	0.62 ms	0.61 ms	0.64 ms
total SQP step	73.37 ms	33.98 ms	8.29 ms	**7.66 ms**

scheme over the standard (MS) method. Note that these timing results include a block based regularization of the Hessian to guarantee a convex structured QP subproblem in the exact Hessian based SQP implementation [268]. A more detailed discussion on how the algorithm is affected by different techniques to perform a sparsity preserving Hessian regularization is outside the scope here, but more information on this topic can be found in [310]. Note that one iteration of the general-purpose sparse NLP solver `Ipopt` on the time optimal OCP with $n_{\mathrm{m}} = 5$ takes about 300 ms in this case, while the solution of one direct collocation based QP takes about 5 s using the sparse `OOQP` solver.

The convergence of the SQP method using the different variants of lifted collocation is illustrated in Figure 6.5 for both the minimum effort and the time optimal OCP formulation. The figure shows the distance $\|W - W^\star\|_\infty$ of the current iterate W from the local minimum $W^\star$ of the direct collocation NLP for the continuous time OCP in Eq. (6.54). Since the exact lifting scheme (LC-EN) is equivalent to direct collocation as shown in Proposition 6.2, it is expected that its convergence is close to that of the standard multiple shooting method (MS) which is also confirmed by the results in Figure 6.5. In addition, the reduction in convergence speed by using a Jacobian approximation in the INIS based lifted collocation integrators appears to be relatively small for this numerical case study. Instead, the adjoint-based IN scheme from Algorithm 6 shows a considerably slower local convergence rate based on the same Jacobian approximation. The latter observation will be clarified as part of Chapter 7.

Table 6.6: Average exact Hessian based SQP timing results for the time optimal chain mass problem using a 4-stage Gauss collocation method ($N_\mathrm{s} = 3, q = 4$), including different numbers of masses n_m and resulting numbers of states n_x.

n_m	n_x	Without lifting (MS)	Exact lifting (LC-EN)	IN lifting (LC-IN)	INIS lifting (LC-INIS)
3	13	18.98 ms	16.4 ms	10.96 ms	8.87 ms
4	19	44.86 ms	30.22 ms	16.26 ms	15.01 ms
5	25	96.77 ms	61.55 ms	25.09 ms	24.92 ms
6	31	169.53 ms	101.56 ms	40.72 ms	39.83 ms
7	37	285.06 ms	157.39 ms	62.40 ms	59.27 ms

Table 6.7: Detailed timing results for exact Hessian based SQP on the time optimal OCP using $n_\mathrm{m} = 5$ masses or $n_\mathrm{x} = 24 + 1$ states ($N_\mathrm{s} = 3, q = 4$). Note that one iteration of direct collocation (6.7) based on `Ipopt` takes about 300 ms, and one sparse QP solution using `OOQP` takes 5 s on average.

	Without lifting (MS)	Exact lifting (LC-EN)	IN lifting (LC-IN)	INIS lifting (LC-INIS)
simulation	**87.23 ms**	51.33 ms	15.50 ms	**15.48 ms**
condensing	2.07 ms	2.08 ms	2.05 ms	2.06 ms
regularization	1.72 ms	1.82 ms	1.86 ms	1.86 ms
QP solution	5.69 ms	6.13 ms	5.67 ms	5.50 ms
total SQP step	96.77 ms	61.55 ms	25.09 ms	**24.92 ms**

6.7 Conclusions and Outlook

This chapter presents a novel family of lifted Newton-type optimization algorithms for direct optimal control, based on collocation within direct multiple shooting. The schemes result in multiple shooting type subproblems, while they all converge locally to the solution of the direct collocation NLP. In case of the exact lifting scheme in Algorithm 5, the iterates are shown to be equivalent to those of a Newton-type optimization method for direct collocation. As we summarized in Table 6.1, the main motivation for lifted collocation is the use of tailored solvers for the multiple shooting type optimal control structure, as well as the possibility to include efficient Newton-type implementations.

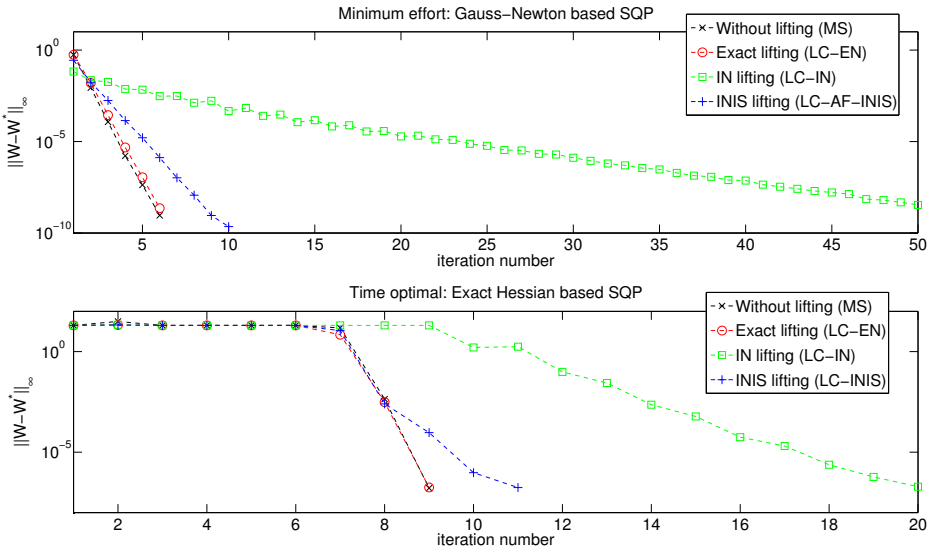


Figure 6.5: SQP convergence using different lifting techniques for the minimum effort (Gauss-Newton) and time optimal OCP (exact Hessian) with $n_{\mathrm{m}} = 5$.

This chapter proposed two types of inexact lifting schemes, using either an adjoint-based implementation in Algorithm 6 or the inexact Newton method with iterated sensitivities in Algorithm 7 and 8. In addition to discussing their implementation as summarized by Table 6.2 and their corresponding properties, a connection has been made to Newton-type convergence theory.

This work includes an open-source software implementation of the proposed algorithms within the `ACADO` code generation tool for real-time optimal control. The performance of the lifted collocation integrators within this package has been illustrated based on the benchmark case study of the optimal control for a chain of masses. Based on these numerical results, a computational speedup of factor 2 is typically observed when using the exact lifting scheme instead of the standard collocation integrator within direct multiple shooting. In addition, a further speedup factor in the range of 5-10 per iteration has been observed when using the inexact Newton based lifted collocation schemes. More elaborate benchmarking of the numerical performance of these lifted collocation integrators for direct optimal control, including pseudospectral methods [126] based on high order collocation polynomials, is part of ongoing research.

Chapter 7

Local Convergence of Inexact Newton with Iterated Sensitivities

Newton-type schemes based on inexact derivatives do not converge to a solution of the original nonlinear optimization problem, unless adjoint derivatives are evaluated in order to compute the correct gradient of the Lagrangian [49, 94]. It has been pointed out by [252] that the locally linear convergence rate of such a standard Inexact Newton (IN) based optimization scheme is not strongly connected to the contraction of the iterations for the inner forward simulation problem. For example, it is possible that the constraint Jacobian approximation results in a fast contraction of the forward problem alone, while the optimization algorithm based on the same Jacobian approximation diverges. In this chapter, we show that the Inexact Newton method with Iterated Sensitivities (INIS) allows one to recover a strong connection between the local contraction rate of the forward problem and the local convergence properties of the resulting optimization algorithm. More specifically, local convergence for the Newton-type method on the forward problem is shown to be necessary, and under mild conditions even sufficient for the asymptotic contraction of the corresponding INIS-type optimization algorithm.

This chapter is largely based on the article in [264], although the results on lifted collocation integrators in the previous chapter form an important motivation for our study on the local convergence of the INIS algorithm.

Outline The chapter is organized as follows. Section 7.1 first briefly presents standard Newton-type optimization for the considered problem formulation. Section 7.2 then proposes and analyzes the Inexact Newton method based on Iterated Sensitivities (INIS) as an alternative implementation of inexact Newton-type optimization. An adjoint-free variant of the INIS-type optimization algorithm is presented in Section 7.3. An important application of the proposed schemes for simultaneous approaches of direct optimal control is discussed in Section 7.4, including numerical results based on the open-source implementation in the `ACADO` code generation tool.

7.1 Problem Formulation

The present chapter considers Newton-type optimization algorithms for a class of nonlinear programming problems

$$\min_{K,w} \quad \psi(K,w) \tag{7.1a}$$

$$\text{s.t.} \quad 0 \;=\; c(K,w), \tag{7.1b}$$

where $K \in \mathbb{R}^{n_\mathrm{K}}$ and $w \in \mathbb{R}^{n_\mathrm{w}}$ are the optimization variables. The objective and constraint functions are defined as $\psi : \mathbb{R}^{n_\mathrm{K}} \times \mathbb{R}^{n_\mathrm{w}} \to \mathbb{R}$ and $c : \mathbb{R}^{n_\mathrm{K}} \times \mathbb{R}^{n_\mathrm{w}} \to \mathbb{R}^{n_c}$ respectively, and are assumed to be twice continuously differentiable in all arguments (see Assumption 1.10). For the sake of simplicity, we omit possibly nonlinear inequality constraints in the NLP (7.1). Note however that our discussion on the local convergence of Newton-type methods can be readily extended to the general case of inequality-constrained optimization.

7.1.1 The Forward Problem

The subset of the variables $K \in \mathbb{R}^{n_\mathrm{K}}$ is selected such that $n_c = n_\mathrm{K}$ and the constraint Jacobian matrix $\frac{\partial c}{\partial K}(\cdot) \in \mathbb{R}^{n_\mathrm{K} \times n_\mathrm{K}}$ is invertible. It follows that these variables are implicitly defined via the nonlinear equality constraints $c(K,w) = 0$. This set of constraints will further be referred to as the *forward problem*, which delivers $K^\star(\bar{w})$ by solving the corresponding system

$$c(K,\bar{w}) = 0, \qquad \text{for a given value } \bar{w}. \tag{7.2}$$

We are interested in solving the forward problem in Eq. (7.2) using Newton-type schemes that do not rely on an exact factorization of $c_K := \frac{\partial c}{\partial K}$, but use instead a full-rank approximation $M \approx c_K$. This Jacobian approximation can be used

directly in a Newton-type method to solve the forward problem by steps

$$\Delta K = -M^{-1} c(\bar{K}, \bar{w}), \tag{7.3}$$

where $\bar{K}$ denotes the current guess and the full-step update in each Newton-type iteration can be written as $\bar{K}^+ = \bar{K} + \Delta K$. Some interesting examples of such problem formulations result from a simultaneous approach to dynamic optimization [34, 51], where the forward problem imposes the system dynamics and therefore corresponds to a numerical simulation of differential equations. A popular approach is direct collocation [39], where the forward problem consists of the collocation equations as discussed in the previous chapter.

7.1.2 Newton-Type Optimization

The Lagrange function for the NLP (7.1) reads as $\mathcal{L}(y, \lambda) = \psi(y) + \lambda^\top c(y)$, where $y := \begin{bmatrix} K^\top & w^\top \end{bmatrix}^\top \in \mathbb{R}^{n_\mathrm{y}}$ denotes all primal variables and $\lambda \in \mathbb{R}^{n_\mathrm{K}}$ denotes the multipliers for the nonlinear equality constraints in Eq. (7.1b). The corresponding first-order necessary conditions for optimality are defined by Theorem 1.16, which can also be written in the compact notation

$$\mathcal{F}(y, \lambda) = \begin{bmatrix} \nabla_y \mathcal{L}(y, \lambda) \\ c(y) \end{bmatrix} = 0. \tag{7.4}$$

Each local minimizer $(y^\star, \lambda^\star)$ of the NLP (7.1) is assumed to be a regular KKT point $\mathcal{F}(y^\star, \lambda^\star) = 0$ as described in Definition 1.20.

Newton-type optimization then proceeds with applying a variant of Newton's method [79, 82] to find a solution to the KKT system in Eq. (7.4). Note that an exact Newton iteration reads as in Eq. (1.15) where $c_y(\bar{y}) := \frac{\partial c}{\partial y}(\bar{y})$ denotes the Jacobian matrix and the values $\bar{y}$ and $\bar{\lambda}$ denote the primal and dual variables at the current guess. In the following, we will refer to the exact Newton iteration using the more compact notation

$$J(\bar{y}, \bar{\lambda}) \begin{bmatrix} \Delta y \\ \Delta \lambda \end{bmatrix} = -\mathcal{F}(\bar{y}, \bar{\lambda}), \tag{7.5}$$

where the exact Jacobian matrix is defined as $J(\bar{y}, \bar{\lambda}) := \frac{\partial \mathcal{F}}{\partial (y, \lambda)}(\bar{y}, \bar{\lambda})$. In this work, a full-step update of the primal and dual variables is considered for simplicity in each iteration, i.e., $\bar{y}^+ = \bar{y} + \Delta y$ and $\bar{\lambda}^+ = \bar{\lambda} + \Delta \lambda$ even though globalization strategies are typically used to guarantee convergence [39, 232]. Newton-type optimization methods have been proposed, which result in desirable local convergence properties at a considerably reduced computational cost by either forming an approximation of the KKT matrix $J(\cdot)$ [37, 181, 322] or by

solving the linear system approximately [73, 74, 162]. For example, the family of Quasi-Newton methods [80, 94, 143, 232] is based on the approximation of the Hessian of the Lagrangian using only first order derivative information. Other Newton-type optimization algorithms even use an inexact Jacobian for the nonlinear constraints [49, 175, 321] as discussed next.

7.1.3 Adjoint-Based Inexact Newton (IN)

Let us consider the Jacobian approximation $M \approx c_K$ in the Newton-type method of Eq. (7.3) to solve the forward problem (7.2). The resulting inexact Newton method aimed at solving the KKT conditions for the NLP in Eq. (7.4), iteratively solves the following linear system

$$\underbrace{\begin{bmatrix} \tilde{H} & \begin{pmatrix} M^\top \\ c_w^\top \end{pmatrix} \\ \begin{pmatrix} M & c_w \end{pmatrix} & \mathbb{0} \end{bmatrix}}_{=: \tilde{J}_{\mathrm{IN}}(\bar{y},\bar{\lambda})} \begin{bmatrix} \Delta K \\ \Delta w \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_y \mathcal{L}(\bar{y},\bar{\lambda}) \\ c(\bar{y}) \end{bmatrix}, \tag{7.6}$$

where an additional approximation of the Hessian $\tilde{H} \approx \nabla_y^2 \mathcal{L}(\bar{y},\bar{\lambda})$ has been introduced for the sake of completeness. Note that the gradient of the Lagrangian $\nabla_y \mathcal{L}(\cdot)$ can be evaluated efficiently using adjoint differentiation techniques, such that the scheme is often referred to as an *adjoint-based Inexact Newton* (IN) method [49, 94]. Each iteration solves the linear system in Eq. (7.6), which can be written in the following compact form

$$\tilde{J}_{\mathrm{IN}}(\bar{y},\bar{\lambda}) \begin{bmatrix} \Delta y \\ \Delta \lambda \end{bmatrix} = -\mathcal{F}(\bar{y},\bar{\lambda}). \tag{7.7}$$

The convergence of this scheme then follows the classical and well-known local contraction result from Theorem 1.26. The simple but restrictive condition in (1.18) on local Newton-type convergence will be sufficient for our discussion, even though more advanced results exist [77, 82, 206]. Algorithm 9 describes a possible implementation to solve the adjoint-based IN system from Eq. (7.6). It relies on a numerical elimination of the variables $\Delta K = -M^{-1}(c(\bar{y}) + c_w \Delta w)$ and $\Delta\lambda$ such that a smaller system is solved in the variables Δw, which can be expanded back into the full variable space. Note that one recovers the Newton-type iteration on the forward problem $\Delta K = -M^{-1} c(\bar{y})$ for a fixed value $\bar{w}$, i.e., when $\Delta w = 0$.

Remark 7.1 *The matrix* $\tilde{Z}^\top := \begin{bmatrix} -c_w^\top M^{-\top}, \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$ *in step 1 of Algorithm 9 is an approximation for* $Z^\top := \begin{bmatrix} -c_w^\top c_K^{-\top}, \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$*, which is defined such that* $Z^\top \nabla_y \mathcal{L}(\bar{y},\bar{\lambda}) = Z^\top \nabla_y \psi(\bar{y})$*. When using instead the approximate matrix* $\tilde{Z}$*,*

this results in the following correction of the gradient term

$$\tilde{Z}^\top \nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) = \tilde{Z}^\top \nabla_y \psi(\bar{y}) - \left(c_K M^{-1} c_w - c_w\right)^\top \bar{\lambda}. \tag{7.8}$$

Algorithm 9 One iteration of an adjoint-based Inexact Newton (IN) method.

Input: Current values $\bar{y} = (\bar{K}, \bar{w})$, $\bar{\lambda}$ and approximations M, $\tilde{H}(\bar{y}, \bar{\lambda})$.

1: After eliminating the variables ΔK, $\Delta\lambda$ in (7.6), solve the resulting system:
$\tilde{Z}^\top \tilde{H} \tilde{Z}\, \Delta w = -\tilde{Z}^\top \left(\nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) + \tilde{H} \begin{bmatrix} -M^{-1} c(\bar{y}) \\ \mathbb{0} \end{bmatrix} \right)$,
where $\tilde{Z}^\top := \left[-c_w^\top M^{-\top},\ \mathbb{1}_{n_\mathrm{w}}\right]$.

2: Based on Δw, the corresponding values for ΔK and $\Delta\lambda$ are found:
$\Delta K = -M^{-1}(c(\bar{y}) + c_w \Delta w)$ and $\Delta\lambda = -\begin{bmatrix} M^{-\top} & \mathbb{0} \end{bmatrix} \left(\nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) + \tilde{H} \Delta y\right)$.

Output: New values $\bar{y}^+ = \bar{y} + \Delta y$ and $\bar{\lambda}^+ = \bar{\lambda} + \Delta\lambda$.

7.1.4 A Motivating QP Example

In this chapter, we are interested in the existence of a connection between the Newton-type iteration on the forward problem (7.3) being locally contractive, i.e., $\kappa_\mathrm{F}^\star := \rho\left(M^{-1} c_K - \mathbb{1}_{n_\mathrm{K}}\right) < 1$, and the local convergence for the corresponding Newton-type optimization algorithm as defined by Theorem 1.26. From the detailed discussion in [45, 252, 253], we know that contraction for the forward problem is neither sufficient nor necessary for convergence of the adjoint-based IN method in Algorithm 9, even when using an exact Hessian $H = \nabla_y^2 \mathcal{L}(\bar{y}, \bar{\lambda})$. To illustrate this statement, let us consider the following QP example from [252] based on a linear constraint $c(y) = \begin{bmatrix} A_1 & A_2 \end{bmatrix} y$ and a quadratic objective $\psi(y) = \frac{1}{2} y^\top H y$ in Eq. (7.1). The matrix A_1 is assumed invertible and close to identity, such that we can select the Jacobian approximation $M = \mathbb{1}_{n_\mathrm{K}} \approx A_1$. The problem data from [252] read as

$$H = \begin{bmatrix} 0.83 & 0.083 & 0.34 & -0.21 \\ 0.083 & 0.4 & -0.34 & -0.4 \\ 0.34 & -0.34 & 0.65 & 0.48 \\ -0.21 & -0.4 & 0.48 & 0.75 \end{bmatrix}, \tag{7.9}$$

$$A_1 = \begin{bmatrix} 1.1 & 1.7 \\ 0 & 0.52 \end{bmatrix},\ A_2 = \begin{bmatrix} -0.55 & -1.4 \\ -0.99 & -1.8 \end{bmatrix}.$$

For this specific QP instance due to Potschka [252], we compute the linear contraction rate for the Newton-type method on the forward problem (7.3):

$$\kappa_{\mathrm{F}}^{\star} = \rho(M^{-1}c_K - \mathbb{1}_{n_{\mathrm{K}}}) = \rho(A_1 - \mathbb{1}_{n_{\mathrm{K}}}) = 0.48 < 1.$$

In addition, let us consider the IN algorithm based on the solution of the linear system (7.6) for the same QP example using the exact Hessian $\tilde{H} = H$. We can compute the corresponding contraction rate

$$\kappa_{\mathrm{IN}}^{\star} = \rho(\tilde{J}_{\mathrm{IN}}^{-1}J - \mathbb{1}) \approx 1.625 > 1,$$

where $J = J(y, \lambda)$ denotes the exact Jacobian of the KKT system in Eq. (7.4). For this QP example (7.9), the Newton-type method on the forward problem locally converges with $\kappa_{\mathrm{F}}^{\star} = 0.48 < 1$, while the corresponding IN algorithm is unstable $\kappa_{\mathrm{IN}}^{\star} \approx 1.625 > 1$. This means that one indeed does not necessarily obtain a converging IN scheme, only by proposing a sufficiently accurate Jacobian approximation for the forward problem. In what follows, we present and study a novel inexact Newton-type optimization algorithm based on iterated sensitivities, which circumvents this problem at a negligible computational cost. These observations are illustrated in Figure 7.1, which presents the Newton-type iterations for the different algorithms starting from the same initial point. The figure includes the linear convergence for the Newton-type method on the forward problem as defined in Eq. (7.3).

7.2 Inexact Newton with Iterated Sensitivities (INIS)

Let us introduce an alternative inexact Newton-type optimization algorithm, labelled INIS in the following, based on the solution of an augmented KKT system defined as

$$\mathcal{F}_{\mathrm{INIS}}(y, \lambda, D) = \begin{bmatrix} \nabla_y \mathcal{L}(y, \lambda) \\ c(y) \\ \mathrm{vec}(c_K D - c_w) \end{bmatrix} = 0, \tag{7.10}$$

where the additional variable $D \in \mathbb{R}^{n_{\mathrm{K}} \times n_{\mathrm{w}}}$ denotes the sensitivity matrix implicitly defined by the equation $c_K D - c_w = 0$ such that $n_{\mathrm{D}} = n_{\mathrm{K}} n_{\mathrm{w}}$[1]. Note that Proposition 6.11 stated the connection between the augmented system $\mathcal{F}_{\mathrm{INIS}}(y, \lambda, D) = 0$ and the original KKT conditions in (7.4).

[1] The operator $\mathrm{vec}(\cdot)$ denotes a vectorization of a matrix, i.e., this is a linear transformation that converts the matrix into a column vector.

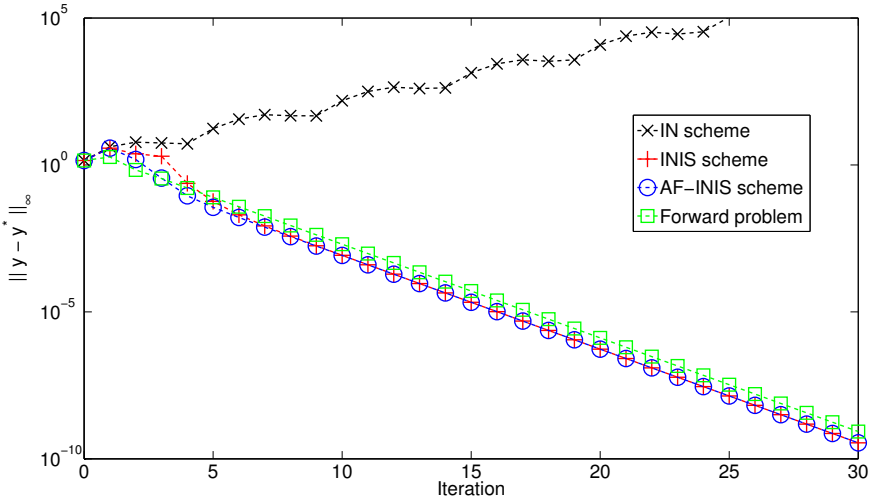


Figure 7.1: Illustration of the divergence of the IN and the convergence of the INIS scheme for the QP in Eq. (7.9). In addition, the asymptotic rate of convergence for INIS can be observed to be the same as for the forward problem.

7.2.1 Implementation INIS

We introduce the Inexact Newton method with Iterated Sensitivities (INIS), to iteratively solve the augmented KKT system (7.10) based on

$$\underbrace{\begin{bmatrix} \tilde{H} & \begin{pmatrix} M^\top \\ \bar{D}^\top M^\top \end{pmatrix} & \mathbb{0} \\ \begin{pmatrix} M & M\,\bar{D} \end{pmatrix} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes M \end{bmatrix}}_{=:\,\tilde{J}_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{D})} \begin{bmatrix} \Delta K \\ \Delta w \\ \Delta\lambda \\ \mathrm{vec}(\Delta D) \end{bmatrix} = - \underbrace{\begin{bmatrix} \nabla_y \mathcal{L}(\bar{y},\bar{\lambda}) \\ c(\bar{y}) \\ \mathrm{vec}(c_K \bar{D} - c_w) \end{bmatrix}}_{=\mathcal{F}_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{D})}, \tag{7.11}$$

where $\otimes$ denotes the Kronecker product of matrices and using the Jacobian approximation $M \approx c_K$ from the Newton-type method on the forward problem in (7.3). The resulting matrix $\tilde{J}_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{D})$ forms an approximation for the exact Jacobian $J_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{D}) := \frac{\partial \mathcal{F}_{\mathrm{INIS}}}{\partial(y,\lambda,D)}(\bar{y},\bar{\lambda},\bar{D})$ of the augmented system. As discussed in the previous chapter, this scheme shows a connection to the lifted Newton method based on a lifting of the matrix D defined by the sensitivity equation $c_K D - c_w = 0$. Similar to the case of the standard lifted Newton method [8], Algorithm 10 shows that the INIS scheme in Eq. (7.11) can be implemented efficiently using a condensing and expansion procedure. The

computational cost of the algorithm can be made close to that of the standard inexact Newton method in Algorithm 9. More specifically, the INIS scheme requires the linear system solution $-M^{-1}(c_K\bar{D} - c_w)$ which can be performed efficiently using AD techniques [141] to evaluate the right-hand side. Similar to Remark 7.1, let us write the gradient correction in step 1 of Algorithm 10:

$$\tilde{Z}^\top \nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) = \tilde{Z}^\top \nabla_y \psi(\bar{y}) - (c_K\bar{D} - c_w)^\top \bar{\lambda}, \tag{7.12}$$

where $\tilde{Z}^\top := \begin{bmatrix} -\bar{D}^\top, \ \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$. Note that the evaluation of $c_K\bar{D} - c_w$ can be reused in step 1 and 3 of Algorithm 10, which allows INIS to be computationally competitive with the standard IN scheme. This will also be illustrated by the numerical results for direct optimal control in Section 7.4.

Algorithm 10 One iteration of an adjoint-based Inexact Newton with Iterated Sensitivities (INIS) optimization method.

Input: Current values $\bar{y} = (\bar{K}, \bar{w})$, $\bar{\lambda}$, $\bar{D}$ and approximations M, $\tilde{H}(\bar{y}, \bar{\lambda})$.

1: After eliminating the variables ΔK, $\Delta\lambda$ in (7.11), solve the resulting system:
$\tilde{Z}^\top \tilde{H} \tilde{Z}\, \Delta w = -\tilde{Z}^\top \left(\nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) + \tilde{H} \begin{bmatrix} -M^{-1} c(\bar{y}) \\ \mathbb{0} \end{bmatrix} \right)$,
where $\tilde{Z}^\top := \begin{bmatrix} -\bar{D}^\top, \ \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$.

2: Based on Δw, the corresponding values for ΔK and $\Delta\lambda$ are found:
$\Delta K = -M^{-1} c(\bar{y}) - \bar{D}\Delta w$ and $\Delta\lambda = -\begin{bmatrix} M^{-\top} & \mathbb{0} \end{bmatrix} \left(\nabla_y \mathcal{L}(\bar{y}, \bar{\lambda}) + \tilde{H} \Delta y \right)$.

3: Independently, the sensitivity matrix is updated in each iteration:
$\Delta D = -M^{-1}(c_K\bar{D} - c_w)$.

Output: New values $\bar{y}^+ = \bar{y} + \Delta y$, $\bar{\lambda}^+ = \bar{\lambda} + \Delta\lambda$ and $\bar{D}^+ = \bar{D} + \Delta D$.

7.2.2 Main Result: Local Contraction Theorem

In what follows, we show that INIS-type optimization allows one to recover the connection between the contraction properties of the forward problem and the Newton-type optimization algorithm. This observation makes the INIS scheme depart fundamentally from the classical adjoint-based IN method. The local contraction of the forward problem will be shown to be necessary for the convergence of the corresponding INIS algorithm, and can be sufficient under reasonable assumptions on the Hessian approximation $\tilde{H}$.

Let us formalize the local contraction rate $\kappa^\star_{\text{INIS}} = \rho(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1})$ for the INIS scheme, where the Jacobian of the augmented KKT system reads

$$J_{\text{INIS}} = \begin{bmatrix} \nabla^2_y \mathcal{L} & c_y^\top & \mathbb{0} \\ c_y & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{1}_{n_\text{w}} \otimes c_K \end{bmatrix} \quad \text{where } s_y := \frac{\partial}{\partial y} \operatorname{vec}(c_K D - c_w). \tag{7.13}$$

The following theorem specifies the eigenspectrum of the iteration matrix $\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}$ at the solution point $(y^\star, \lambda^\star, D^\star)$, using the notation $\sigma(P)$ to denote the set of eigenvalues for a matrix P.

Theorem 7.2 (Eigenspectrum INIS iteration matrix) *For the augmented linear system* (7.11) *on the NLP in Eq.* (7.1)*, the eigenspectrum of the INIS-type iteration matrix at the solution* $(y^\star, \lambda^\star, D^\star)$ *reads as*

$$\sigma\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}\right) = \sigma\left(M^{-1} c_K - \mathbb{1}_{n_\text{K}}\right) \cup \sigma\left(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_\text{w}}\right), \tag{7.14}$$

where $Z^\top := \left[-c_w^\top c_K^{-\top},\ \mathbb{1}_{n_\text{w}}\right]$ *denotes a basis for the null space of the constraint Jacobian* $c_y Z = \mathbb{0}$*, such that the reduced Hessians* $H_z := Z^\top H Z$ *and* $\tilde{H}_z := Z^\top \tilde{H} Z \in \mathbb{R}^{n_\text{w} \times n_\text{w}}$ *are defined. Note that* $H := \nabla^2_y \mathcal{L}(y^\star, \lambda^\star)$ *is the exact Hessian and* $\tilde{H} \approx H$ *is an approximation. More specifically, the iteration matrix has the* n_w *eigenvalues of the matrix* $\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_\text{w}}$ *and the* n_K *eigenvalues of* $M^{-1} c_K - \mathbb{1}_{n_\text{K}}$ *with an algebraic multiplicity of* $2 + n_\text{w}$*.*

Proof. At the solution of the augmented KKT system in Eq. (7.10), the sensitivity matrix corresponds to $D^\star = c_K^{-1} c_w$. We then introduce the following Jacobian matrix and its approximation:

$$c_y = \begin{bmatrix} c_K & c_w \end{bmatrix}, \qquad \tilde{c}_y = \begin{bmatrix} \mathbb{1}_{n_\text{K}} & D^\star \end{bmatrix} = c_K^{-1} c_y,$$

such that the exact and inexact augmented Jacobian matrices read

$$J_{\text{INIS}} = \begin{bmatrix} H & c_y^\top & \mathbb{0} \\ c_y & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{1}_{n_\text{w}} \otimes c_K \end{bmatrix}, \qquad \tilde{J}_{\text{INIS}} = \begin{bmatrix} \tilde{H} & \tilde{c}_y^\top M^\top & \mathbb{0} \\ M \tilde{c}_y & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\text{w}} \otimes M \end{bmatrix}, \tag{7.15}$$

at the solution point $(y^\star, \lambda^\star, D^\star)$. We observe that the eigenvalues γ of the iteration matrix $\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}$ are given by

$$\det\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1} - \gamma \mathbb{1}\right) = \det\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - (\gamma + 1)\mathbb{1}\right) = 0.$$

Since $\tilde{J}_{\text{INIS}}$ is invertible, this equality holds if and only if

$$\det\left(\tilde{J}_{\text{INIS}}\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - (\gamma + 1)\mathbb{1}\right)\right) = \det\left(J_{\text{INIS}} - (\gamma + 1)\tilde{J}_{\text{INIS}}\right) = 0.$$

Using the notation in Eq. (7.15), we can rewrite the matrix $J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}$ as the following product of block matrices:

$$J_{\text{INIS}} - (\gamma+1)\,\tilde{J}_{\text{INIS}} =$$

$$\begin{bmatrix} \mathbb{1}_{n_\text{y}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\text{D}} \end{bmatrix} \begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top & \mathbb{0} \\ \tilde{c}_y & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{1}_{n_\text{w}} \otimes \tilde{M} \end{bmatrix} \begin{bmatrix} \mathbb{1}_{n_\text{y}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\text{D}} \end{bmatrix}^\top \tag{7.16}$$

where the matrix $\tilde{M} = c_K - (\gamma+1)\,M$ is defined such that $\tilde{M}\tilde{c}_y = c_y - (\gamma+1)\,M\,\tilde{c}_y$. The determinant of the product of matrices in Eq. (7.16) can be rewritten as

$$\begin{aligned} &\det\left(J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}\right) \\ &= \det\left(\begin{bmatrix} \mathbb{1}_{n_\text{y}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\text{D}} \end{bmatrix}\right)^2 \det\left(\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top & \mathbb{0} \\ \tilde{c}_y & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{1}_{n_\text{w}} \otimes \tilde{M} \end{bmatrix}\right) \\ &= \det\left(\tilde{M}\right)^{2+n_\text{w}} \det\left(\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top \\ \tilde{c}_y & \mathbb{0} \end{bmatrix}\right). \end{aligned} \tag{7.17}$$

Note that the Jacobian approximation M is invertible such that the determinant $\det(\tilde{M})$ is zero if and only if $\det\left(M^{-1}c_K - (\gamma+1)\,\mathbb{1}_{n_\text{K}}\right) = 0$. It follows that $\det\left(J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}\right) = 0$ holds for only the values of γ that fulfill the conditions

$$\det\left(M^{-1}c_K - (\gamma+1)\,\mathbb{1}_{n_\text{K}}\right) = 0, \qquad \text{or} \tag{7.18a}$$

$$\det\left(\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top \\ \tilde{c}_y & \mathbb{0} \end{bmatrix}\right) = 0. \tag{7.18b}$$

Note that (7.18a) is satisfied for only the eigenvalues $\gamma \in \sigma\left(M^{-1}c_K - \mathbb{1}_{n_\text{K}}\right)$ with an algebraic multiplicity $n_\text{w}+2$ as can be observed directly in Eq. (7.17). It can be verified that the values for γ satisfying (7.18b) are given by:

$$\det\left(Z^\top\left(H - (\gamma+1)\,\tilde{H}\right)Z\right) = \det\left(H_z - (\gamma+1)\,\tilde{H}_z\right) = 0, \tag{7.19}$$

where $Z = \begin{bmatrix} -c_K^{-1}c_w \\ \mathbb{1}_{n_\text{w}} \end{bmatrix}$ denotes a basis for the null space of the constraint Jacobian such that $c_y Z = \mathbb{0}$ or $\tilde{c}_y Z = \mathbb{0}$. The last equality in (7.19) is satisfied for only the eigenvalues $\gamma \in \sigma\left(H_z^{-1}H_z - \mathbb{1}_{n_\text{w}}\right)$. Note that this corresponds to an eigenvalue $\gamma = 0$ in the case of an exact Hessian matrix $\tilde{H} = H$. □

Based on these results regarding the eigenspectrum of the iteration matrix, we now formally state the local contraction theorem for the INIS method.

Corollary 7.3 (Local INIS-type contraction) *The local rate of convergence for the INIS-type optimization algorithm is defined by*

$$\kappa^\star_{\text{INIS}} = \rho\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}\right) = \max\left(\kappa^\star_{\text{F}},\, \rho(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_{\text{w}}})\right),$$

where $\kappa^\star_{\text{F}} = \rho(M^{-1} c_K - \mathbb{1}_{n_{\text{K}}})$ *is defined for the Newton-type method on the forward problem in* (7.3)*. It follows that local contraction for the forward problem, i.e.,* $\kappa^\star_{\text{F}} < 1$*, is necessary for local convergence of the INIS-type algorithm. Under the condition* $\rho(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_{\text{w}}}) \leq \kappa^\star_{\text{F}}$ *on the quality of the Hessian approximation, e.g.,* $\rho(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_{\text{w}}}) = 0$ *in case of an exact Hessian, local contraction for the forward problem is additionally sufficient since the asymptotic rate of convergence satisfies* $\kappa^\star_{\text{INIS}} = \kappa^\star_{\text{F}}$*.*

7.2.3 Numerical Examples

We first revisit the motivating QP example from Section 7.1.4, where the asymptotic contraction rate for the Newton-type method on the forward problem reads $\kappa^\star_{\text{F}} = 0.48 < 1$. In contrast, the solution was found to be asymptotically unstable since $\kappa^\star_{\text{IN}} \approx 1.625 > 1$ for the IN method based on the same Jacobian approximation. Let us now consider the proposed INIS scheme from Algorithm 10 for the same QP example using the exact Hessian $\tilde{H} = H$. We compute the corresponding contraction rate at the solution

$$\kappa^\star_{\text{INIS}} = \rho(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}) = 0.48 < 1,$$

where J_{INIS} denotes the exact Jacobian of the augmented KKT system in (7.13). Therefore, the INIS scheme indeed exhibits a linear local convergence with the same asymptotic rate as the forward problem, i.e., $\kappa^\star_{\text{INIS}} = 0.48 = \kappa^\star_{\text{F}}$. This result is consistent with Theorem 7.2 and is illustrated in Figure 7.1.

In addition, we introduce a simple example of an NLP (7.1) based on the QP formulation above. For this purpose, let us take a quadratic objective $\psi(y) = \frac{1}{2} y^\top H y + h^\top y$ where H is defined in Eq. (7.9), the gradient vector $h = \begin{bmatrix} 0.1 & 0 & 0 & 0 \end{bmatrix}^\top$ and the nonlinear constraint function reads

$$c(y) = \begin{bmatrix} A_1 & A_2 \end{bmatrix} y + 0.1 \begin{bmatrix} y_1^3 \\ y_2 y_4 \end{bmatrix}, \tag{7.20}$$

where also the matrices A_1 and A_2 are adopted from Eq. (7.9). Figure 7.2 then illustrates the convergence results for the IN and INIS schemes from Algorithm 9

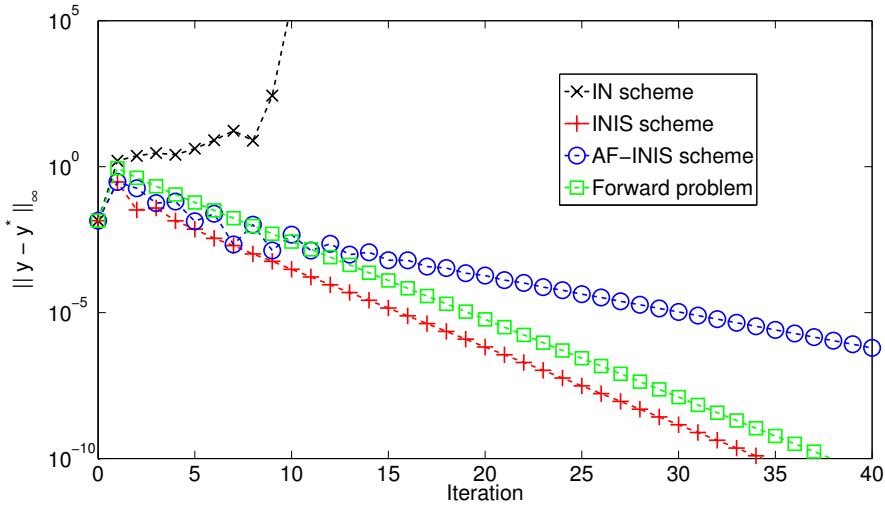


Figure 7.2: Illustration of the divergence of the IN and the convergence of the INIS scheme for the NLP in Eq. (7.20). In addition, the asymptotic rate of convergence for INIS can be observed to be the same as for the forward problem unlike the adjoint-free AF-INIS implementation for this NLP example.

and 10 on this NLP example. It can be observed that the local contraction rate for INIS corresponds to that for the forward problem, while the standard IN implementation locally diverges for this particular example. More specifically, the asymptotic contraction rates at the solution can be computed to be

$$\kappa_{\mathrm{F}}^\star = \kappa_{\mathrm{INIS}}^\star \approx 0.541 < 1 < 1.441 \approx \kappa_{\mathrm{IN}}^\star.$$

7.2.4 Additional Constraints

As mentioned already in Chapter 1, an inequality constrained problem could be solved with any of the proposed Newton-type optimization algorithms, in combination with techniques from either SQP or interior point methods to treat the inequality constraints. Let us consider a local minimizer which is assumed to be regular, i.e., it satisfies the linear independence constraint qualification (LICQ), the strict complementarity condition and the second-order sufficient conditions (SOSC) as in Definition 1.20. In this case, the primal-dual central path associated with this minimizer is locally unique when using an interior point method [232]. In case of an SQP method, under mild conditions

on the Hessian and Jacobian approximations, Theorem 1.28 states that the corresponding active set is locally stable in a neighborhood of the minimizer, i.e., the solution of each QP subproblem has the same active set as the original NLP [49, 52, 284]. Based on these observations, we can readily extend the local contraction theorem for the INIS method to a more general setting, using the following equality constrained NLP formulation

$$\min_{K,w} \quad \psi(K,w) \tag{7.21a}$$

$$\text{s.t.} \quad 0 \;=\; c(K,w), \tag{7.21b}$$

$$0 \;=\; h(K,w). \tag{7.21c}$$

This problem is similar to the NLP in Eq. (7.1) with the additional constraint function $h : \mathbb{R}^{n_\mathrm{K}} \times \mathbb{R}^{n_\mathrm{w}} \to \mathbb{R}^{n_h}$, which is also assumed to be twice continuously differentiable in all arguments. We further let $\nu \in \mathbb{R}^{n_h}$ denote the multipliers for the nonlinear constraints in Eq. (7.21c). For the purpose of a local convergence analysis, note that these equality constraints could additionally comprise the active inequality constraints at the local minimizer $(y^\star, \lambda^\star, \nu^\star)$.

Let us write the INIS iteration from Eq. (7.11) for this NLP as follows

$$\underbrace{\begin{bmatrix} \tilde{H} & \begin{pmatrix} M^\top & h_K^\top \\ \bar{D}^\top M^\top & h_w^\top \end{pmatrix} & \mathbb{0} \\ \begin{pmatrix} M & M\bar{D} \\ h_K & h_w \end{pmatrix} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes M \end{bmatrix}}_{=: \tilde{J}_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{\nu},\bar{D})} \begin{bmatrix} \Delta K \\ \Delta w \\ \Delta\lambda \\ \Delta\nu \\ \mathrm{vec}(\Delta D) \end{bmatrix} = - \underbrace{\begin{bmatrix} \nabla_y \mathcal{L}(\bar{y},\bar{\lambda},\bar{\nu}) \\ c(\bar{y}) \\ h(\bar{y}) \\ \mathrm{vec}(c_K \bar{D} - c_w) \end{bmatrix}}_{= \mathcal{F}_{\mathrm{INIS}}(\bar{y},\bar{\lambda},\bar{\nu},\bar{D})}, \tag{7.22}$$

where the Lagrange function $\mathcal{L}(y,\lambda,\nu) = \psi(y) + \lambda^\top c(y) + \nu^\top h(y)$ is defined and based on a Hessian approximation $\tilde{H} \approx \nabla_y^2 \mathcal{L}(\bar{y},\bar{\lambda},\bar{\nu})$ and the Jacobian approximation $M \approx c_K$ from the Newton-type forward scheme in (7.3). Let us present an extension of the local contraction theorem for the proposed INIS optimization algorithm.

Theorem 7.4 *For the augmented linear system* (7.22) *on the NLP in Eq.* (7.21)*, the eigenspectrum of the INIS iteration matrix at the solution* $(y^\star, \lambda^\star, \nu^\star, D^\star)$ *reads as*

$$\sigma\left(\tilde{J}_{\mathrm{INIS}}^{-1} J_{\mathrm{INIS}} - \mathbb{1}\right) = \{0\} \cup \sigma\left(M^{-1} c_K - \mathbb{1}_{n_\mathrm{K}}\right) \cup \sigma\left(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_z}\right), \tag{7.23}$$

where $n_z = n_\mathrm{w} - n_h$ *and* Z *denotes a basis for the null space of the constraint Jacobian* $\begin{bmatrix} c_y^\top & h_y^\top \end{bmatrix}^\top$*, such that the reduced Hessians* $H_z := Z^\top H Z \in \mathbb{R}^{n_z \times n_z}$

and $\tilde{H}_z := Z^\top \tilde{H} Z \in \mathbb{R}^{n_z \times n_z}$ *are defined. The local rate of convergence for the INIS optimization algorithm then reads as*

$$\kappa^\star_{\text{INIS}} = \rho\left(\tilde{J}^{-1}_{\text{INIS}} J_{\text{INIS}} - \mathbb{1}\right) = \max\left(\kappa^\star_{\text{F}},\, \rho(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_z})\right),$$

where $\kappa^\star_{\text{F}} = \rho(M^{-1} c_K - \mathbb{1}_{n_{\text{K}}})$ *is defined for the Newton-type scheme in* (7.3)*.*

Proof. At the solution of the augmented KKT system for the NLP in Eq. (7.21), the sensitivity matrix corresponds to $D^\star = c_K^{-1} c_w$. We then introduce the exact and inexact augmented Jacobian matrices as

$$J_{\text{INIS}} = \begin{bmatrix} H & c_y^\top & h_y^\top & \mathbb{0} \\ c_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ h_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{w}}} \otimes c_K \end{bmatrix},\ \tilde{J}_{\text{INIS}} = \begin{bmatrix} \tilde{H} & \tilde{c}_y^\top M^\top & h_y^\top & \mathbb{0} \\ M\,\tilde{c}_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ h_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{w}}} \otimes M \end{bmatrix}, \tag{7.24}$$

where $\tilde{c}_y = c_K^{-1} c_y$ is defined at the solution point $(y^\star, \lambda^\star, \nu^\star, D^\star)$. Similar to the proof of Theorem 7.2, we can rewrite $J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}$ as the following product of block matrices:

$$J_{\text{INIS}} - (\gamma+1)\,\tilde{J}_{\text{INIS}} = \begin{bmatrix} \mathbb{1}_{n_y} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & -\gamma\,\mathbb{1}_{n_h} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{D}}} \end{bmatrix}$$

$$\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top & h_y^\top & \mathbb{0} \\ \tilde{c}_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ h_y & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{w}}} \otimes \tilde{M} \end{bmatrix} \begin{bmatrix} \mathbb{1}_{n_y} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & -\gamma\,\mathbb{1}_{n_h} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\text{D}}} \end{bmatrix}^\top, \tag{7.25}$$

where the matrix $\tilde{M} = c_K - (\gamma+1)\,M$ is defined, such that $\tilde{M}\tilde{c}_y = c_y - (\gamma+1)\,M\,\tilde{c}_y$. The determinant of this product of matrices can be rewritten as

$$\begin{aligned} &\det\left(J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}\right) \\ &= (-\gamma)^{2\,n_h} \det\left(\tilde{M}\right)^{2+n_{\text{w}}} \det\left(\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{c}_y^\top & h_y^\top \\ \tilde{c}_y & \mathbb{0} & \mathbb{0} \\ h_y & \mathbb{0} & \mathbb{0} \end{bmatrix}\right). \end{aligned} \tag{7.26}$$

It follows that $\det\left(J_{\text{INIS}} - (\gamma+1)\tilde{J}_{\text{INIS}}\right) = 0$ holds for only the value $\gamma = 0$, the eigenvalues $\gamma \in \sigma\left(M^{-1} c_K - \mathbb{1}_{n_{\text{K}}}\right)$ and the values for γ that satisfy

$$\det\left(Z^\top \left(H - (\gamma+1)\,\tilde{H}\right) Z\right) = \det\left(H_z - (\gamma+1)\,\tilde{H}_z\right) = 0, \tag{7.27}$$

where Z denotes a basis for the null space of the Jacobian matrix for all the equality constraints, i.e., $\begin{bmatrix} c_y \\ h_y \end{bmatrix} Z = \mathbb{0}$. □

In what follows, for simplicity of notation, we restrict to the original NLP formulation of Eq. (7.1). However, we have shown here that these local contraction results can be readily extended further to optimization problems with additional constraints.

7.3 Adjoint-Free INIS Optimization

Algorithm 10 presented the INIS scheme to solve the augmented KKT system in Eq. (7.10), based on adjoint sensitivity propagation to evaluate the gradient of the Lagrangian $\nabla_y \mathcal{L}(y, \lambda) = \nabla_y \psi(y) + \nabla_y c(y)\lambda$. When the constraint function $c(y)$ consists of a long chain of function evaluations, e.g., as is often the case for dynamic optimization, the computation of adjoint derivatives typically results in relatively high storage requirements of the forward variables as discussed in [141]. Unlike the standard IN method in Algorithm 9 for which adjoint sensitivity propagation is necessary for convergence as discussed in [49, 94], the proposed INIS algorithm allows for deploying an adjoint-free implementation as will be presented in this section. For this purpose, note that also a multiplier-free Hessian approximation $\tilde{H}(y) \approx H(y, \lambda) := \nabla_y^2 \mathcal{L}(y, \lambda)$ is required, e.g., based on the Gauss-Newton method [48, 232].

Algorithm 11 One iteration of an Adjoint-Free INIS (AF-INIS) method.

Input: Current values $\bar{y} = (\bar{K}, \bar{w})$, $\bar{D}$ and approximations M, $\tilde{H}(\bar{y})$.

1: After eliminating the variables ΔK, $\Delta\lambda$ in (7.29), solve the resulting system:
$\tilde{Z}^\top \tilde{H} \tilde{Z}\, \Delta w = -\tilde{Z}^\top \left(\nabla_y \psi(\bar{y}) + \tilde{H} \begin{bmatrix} -M^{-1} c(\bar{y}) \\ \mathbb{0} \end{bmatrix} \right)$,
where $\tilde{Z}^\top := \left[-\bar{D}^\top,\ \mathbb{1}_{n_w} \right]$.

2: Based on Δw, the corresponding value for ΔK is found:
$\Delta K = -M^{-1} c(\bar{y}) - \bar{D} \Delta w$.

3: Independently, the sensitivity matrix is updated in each iteration:
$\Delta D = -M^{-1}(c_K \bar{D} - c_w)$.

Output: New values $\bar{y}^+ = \bar{y} + \Delta y$ and $\bar{D}^+ = \bar{D} + \Delta D$.

7.3.1 Implementation AF-INIS

Algorithm 11 presents the adjoint-free variant of the INIS optimization method from Algorithm 10. It corresponds to solving the following approximate variant of the augmented KKT system in Eq. (7.10):

$$\mathcal{F}_{\mathrm{AF}}(y, \lambda, D) = \begin{bmatrix} \nabla_y \psi(y) + \begin{pmatrix} c_K^\top \\ D^\top c_K^\top \end{pmatrix} \lambda \\ c(y) \\ \mathrm{vec}(c_K D - c_w) \end{bmatrix} = 0. \tag{7.28}$$

Note that Proposition 6.12 formalizes the connection between this augmented system of equations and the original NLP in Eq. (7.1). The adjoint-free inexact Newton method with iterated sensitivities (AF-INIS) then uses the same approximate Jacobian matrix $\tilde{J}_{\mathrm{INIS}}(\bar{y}, \bar{\lambda}, \bar{D})$ from Eq. (7.11) to solve the augmented set of equations in (7.28). At each iteration, the corresponding linear system reads as

$$\underbrace{\begin{bmatrix} \tilde{H} & \begin{pmatrix} M^\top \\ \bar{D}^\top M^\top \end{pmatrix} & \mathbb{0} \\ \begin{pmatrix} M & M\bar{D} \end{pmatrix} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes M \end{bmatrix}}_{= \tilde{J}_{\mathrm{INIS}}(\bar{y}, \bar{\lambda}, \bar{D})} \begin{bmatrix} \Delta K \\ \Delta w \\ \Delta \lambda \\ \mathrm{vec}(\Delta D) \end{bmatrix} = - \underbrace{\begin{bmatrix} \nabla_y \psi(\bar{y}) + \begin{pmatrix} c_K^\top \\ \bar{D}^\top c_K^\top \end{pmatrix} \bar{\lambda} \\ c(\bar{y}) \\ \mathrm{vec}(c_K \bar{D} - c_w) \end{bmatrix}}_{= \mathcal{F}_{\mathrm{AF}}(\bar{y}, \bar{\lambda}, \bar{D})}. \tag{7.29}$$

Using this augmented linear system, the steps ΔK, Δw and ΔD can be computed without evaluating any adjoint derivatives as detailed in Algorithm 11. The evaluation of the adjoint derivatives can be avoided because the following term vanishes when multiplying the first equation in the right-hand side of the linear system by $\tilde{Z}^\top := \begin{bmatrix} -\bar{D}^\top & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$:

$$\begin{bmatrix} -\bar{D}^\top & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix} \begin{pmatrix} c_K^\top \\ \bar{D}^\top c_K^\top \end{pmatrix} = -\bar{D}^\top c_K^\top + \bar{D}^\top c_K^\top = \mathbb{0},$$

resulting in an adjoint-free and multiplier-free computation in Algorithm 11. Because we assumed the Hessian approximation $\tilde{H}(\bar{y})$ in this case to be independent of the current multiplier values, we can additionally omit the computation of the update $\Delta\lambda$.

7.3.2 Local Convergence Results

Proposition 6.12 states that, if it converges, the adjoint-free implementation of the INIS method in Algorithm 11 converges to a local minimizer for the NLP

in Eq. (7.1), and this unlike standard adjoint-free inexact Newton methods as discussed in [49, 94]. Even though we will show that the result in Theorem 7.2 does not necessarily hold for the AF-INIS scheme applied to general NLPs, the following theorem extends this local contraction result specifically for QP problems. For this purpose, we introduce the exact Jacobian of the adjoint-free augmented KKT system in Eq. (7.28):

$$J_{\mathrm{AF}}(y,\lambda,D) = \begin{bmatrix} \psi_{yy} + \tilde{c}_{yy} & \begin{pmatrix} c_K^\top \\ D^\top c_K^\top \end{pmatrix} & \begin{pmatrix} \mathbb{0} \\ \mathbb{1}_{n_\mathrm{w}} \otimes \lambda^\top c_K \end{pmatrix} \\ \begin{pmatrix} c_K & c_w \end{pmatrix} & \mathbb{0} & \mathbb{0} \\ s_y & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes c_K \end{bmatrix}, \tag{7.30}$$

where the additional contributions $\psi_{yy} := \nabla_y^2 \psi(y)$ and $\tilde{c}_{yy} := \frac{\partial}{\partial y} \begin{pmatrix} c_K^\top \lambda \\ D^\top c_K^\top \lambda \end{pmatrix}$ are defined and $s_y := \frac{\partial}{\partial y} \operatorname{vec}(c_K D - c_w)$ similar to Eq. (7.13).

Theorem 7.5 (Local AF-INIS contraction) *Let us consider a QP of the form in Eq.* (7.1)*, where the linear constraint* $c(y) = \begin{bmatrix} A_1 & A_2 \end{bmatrix} y$ *and quadratic objective* $\psi(y) = \frac{1}{2} y^\top H y$ *are defined. For the adjoint-free augmented linear system* (7.29) *corresponding to this QP, the eigenspectrum of the AF-INIS iteration matrix reads*

$$\sigma\left(\tilde{J}_{\mathrm{INIS}}^{-1} J_{\mathrm{AF}} - \mathbb{1}\right) = \sigma\left(M^{-1} A_1 - \mathbb{1}_{n_\mathrm{K}}\right) \cup \sigma\left(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_\mathrm{w}}\right), \tag{7.31}$$

at the solution $(y^\star, \lambda^\star, D^\star)$*. The exact Jacobian* $J_{\mathrm{AF}}(y,\lambda,D)$ *is defined by Eq.* (7.30) *for which* $s_y = \mathbb{0}$*,* $\tilde{c}_{yy} = \mathbb{0}$ *and* $\psi_{yy} = H$ *in the case of a QP. Similar to Theorem 7.2, the matrix* $Z^\top := \begin{bmatrix} -A_2^\top A_1^{-\top}, & \mathbb{1}_{n_\mathrm{w}} \end{bmatrix}$ *such that* $H_z := Z^\top H Z \in \mathbb{R}^{n_\mathrm{w} \times n_\mathrm{w}}$ *and* $\tilde{H}_z := Z^\top \tilde{H} Z \in \mathbb{R}^{n_\mathrm{w} \times n_\mathrm{w}}$*. The local rate of convergence for the adjoint-free INIS scheme on the QP is then defined by*

$$\kappa_{\mathrm{AF}}^\star = \rho\left(\tilde{J}_{\mathrm{INIS}}^{-1} J_{\mathrm{AF}} - \mathbb{1}\right) = \max\left(\kappa_{\mathrm{F}}^\star,\, \rho(\tilde{H}_z^{-1} H_z - \mathbb{1}_{n_\mathrm{w}})\right).$$

Proof. At the solution of the adjoint-free augmented KKT system in Eq. (7.28) for the QP formulation, we know that $D^\star = A_1^{-1} A_2$ and we use the notation $A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$ and $\tilde{A} = A_1^{-1} A$. The eigenvalues γ of the iteration matrix $\tilde{J}_{\mathrm{INIS}}^{-1} J_{\mathrm{AF}} - \mathbb{1}$ are given by the expression $\det\left(J_{\mathrm{AF}} - (\gamma + 1)\tilde{J}_{\mathrm{INIS}}\right) = 0$ based on the exact and inexact adjoint-free augmented Jacobian matrices

$$J_{\mathrm{AF}} = \begin{bmatrix} H & A^\top & \begin{pmatrix} \mathbb{0} \\ \mathbb{1}_{n_\mathrm{w}} \otimes {\lambda^\star}^\top A_1 \end{pmatrix} \\ A & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes A_1 \end{bmatrix}, \quad \tilde{J}_{\mathrm{INIS}} = \begin{bmatrix} \tilde{H} & \tilde{A}^\top M^\top & \mathbb{0} \\ M\tilde{A} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_\mathrm{w}} \otimes M \end{bmatrix}, \tag{7.32}$$

at the solution point $(y^\star, \lambda^\star, D^\star)$. We can rewrite $J_{\mathrm{AF}} - (\gamma+1)\tilde{J}_{\mathrm{INIS}}$ as the following product of block matrices:

$$J_{\mathrm{AF}} - (\gamma+1)\,\tilde{J}_{\mathrm{INIS}} =$$

$$\begin{bmatrix} \mathbb{1}_{n_{\mathrm{y}}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\mathrm{D}}} \end{bmatrix} \begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{A}^\top & \begin{pmatrix} \mathbb{0} \\ \mathbb{1}_{n_{\mathrm{w}}} \otimes {\lambda^\star}^\top A_1 \end{pmatrix} \\ \tilde{A} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\mathrm{w}}} \otimes \tilde{M} \end{bmatrix} \begin{bmatrix} \mathbb{1}_{n_{\mathrm{y}}} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{M} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \mathbb{1}_{n_{\mathrm{D}}} \end{bmatrix}^\top, \tag{7.33}$$

where $\tilde{M} = A_1 - (\gamma+1)\,M$. The proof continues in the same way as for Theorem 7.2, since the determinant of the iteration matrix can be written as

$$\begin{aligned} &\det\left(J_{\mathrm{AF}} - (\gamma+1)\tilde{J}_{\mathrm{INIS}}\right) \\ &= \det\left(\tilde{M}\right)^{2+n_{\mathrm{w}}} \det\left(\begin{bmatrix} H - (\gamma+1)\,\tilde{H} & \tilde{A}^\top \\ \tilde{A} & \mathbb{0} \end{bmatrix}\right). \end{aligned} \tag{7.34}$$

□

Remark 7.6 *When applying the adjoint-free INIS scheme from Algorithm 11 to the NLP formulation in Eq.* (7.1)*, the augmented system introduces off-diagonal blocks for the Jacobian matrix as defined in Eq.* (7.30)*. Therefore, the local contraction result in Theorem 7.5 cannot be directly extended to the general NLP case, even though the practical convergence of AF-INIS can typically be expected to be similar for relatively mild nonlinearities in the problem formulation. Note that Figure 7.1 already illustrated the local convergence of the AF-INIS scheme on the QP in* (7.9)*, where* $\kappa^\star_{\mathrm{F}} = \kappa^\star_{\mathrm{INIS}} = \kappa^\star_{\mathrm{AF}} = 0.48 < 1 < 1.625 \approx \kappa^\star_{\mathrm{IN}}$ *holds in that case. For the NLP example in Section 7.2.3, it can be observed from Figure 7.2 that the local convergence rate of AF-INIS is different from that of the INIS scheme, i.e.,* $\kappa^\star_{\mathrm{F}} = \kappa^\star_{\mathrm{INIS}} \approx 0.541 < \kappa^\star_{\mathrm{AF}} \approx 0.753 < 1 < 1.441 \approx \kappa^\star_{\mathrm{IN}}$*, even though it still outperforms the standard IN method.*

7.4 Numerical Optimal Control Results

The previous chapter already motivated the practical applicability of the INIS-type optimization method, either with or without adjoint computation respectively in Algorithm 7 or 10 and Algorithm 8 or 11. Let us briefly present some numerical results based on the chain mass optimal control case study from Section 6.6, in order to illustrate the new findings on the local convergence properties of the INIS optimization scheme.

Table 7.1: Average timing results per Gauss-Newton based SQP iteration on the chain mass optimal control problem using direct collocation ($N_s = 3, q = 4$), including different numbers of masses n_m and states n_x.

n_m	n_x	Gauss-Newton	IN	INIS	AF-INIS
3	12	5.33 ms	2.40 ms	2.19 ms	1.95 ms
4	18	14.79 ms	5.43 ms	4.76 ms	4.29 ms
5	24	34.04 ms	10.71 ms	9.39 ms	7.96 ms
6	30	62.08 ms	18.73 ms	14.88 ms	12.71 ms
7	36	106.57 ms	36.09 ms	21.93 ms	20.06 ms

Table 7.1 presents average timing results per Gauss-Newton based SQP iteration using the `ACADO` code generation tool, and this for different numbers of masses n_m in the minimum effort OCP formulation [2]. Note that the IN, INIS and AF-INIS schemes correspond to the presented implementations in Algorithm 9, 10 and 11. On the other hand, the exact Gauss-Newton method in this case is based on a direct solution of the QP subproblem, corresponding to the linearized KKT conditions including a Gauss-Newton Hessian approximation. The table shows that the use of inexact Jacobian approximations tailored for collocation methods can considerably reduce the computational effort over an exact implementation. A speedup factor of about 5 can be observed here for the INIS scheme, for example, using a Single Newton iteration (see Section 2.4).

Figure 7.3 shows the convergence results for the SQP method, based on these different Newton-type optimization techniques. The figure shows a simulation result for which the inexact Newton (IN) scheme still results in local convergence, even though the contraction rate can be observed to be considerably slower than that for both variants of the proposed INIS algorithm. Note that the Gauss-Newton based Hessian approximation does not depend on the multipliers for the equality constraints, but the convergence of the adjoint-based INIS scheme in Algorithm 10 does depend on the initialization of these Lagrange multipliers unlike the adjoint-free (AF-INIS) variant. This difference in convergence behaviour can also be observed in Figure 7.3. Even though the convergence for both INIS-type variants is close to that for the Newton-type method on the forward problem of this example, the contraction result in Theorem 7.2 cannot generally be extended to the AF-INIS algorithm for nonlinear optimization as discussed earlier in Section 7.3.2. Note that the results for the exact Gauss-Newton method have been included mainly as a reference. It induces a relatively

[2] All numerical simulations are carried out on a standard computer, equipped with an Intel i7-3720QM processor, using a 64-bit version of Ubuntu 14.04 and the g++ 4.8.4 compiler.

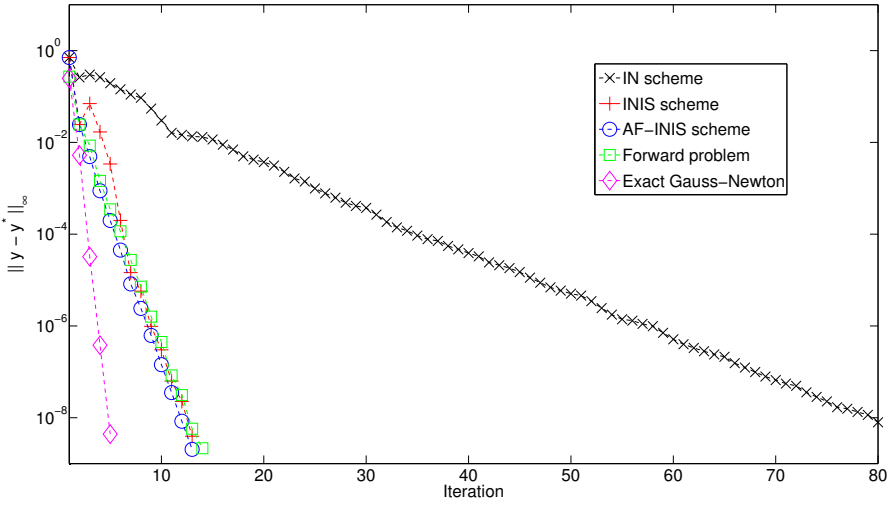


Figure 7.3: Convergence results of the Gauss-Newton based SQP method with different inexact Newton-type techniques for the chain mass optimal control problem using $n_{\mathrm{m}} = 4$ masses.

high computational cost as illustrated by Table 7.1, especially if only rather low accuracy results are sufficient.

7.5 Conclusions and Outlook

This chapter presented a novel family of algorithms for a general class of nonlinear optimization problems, based on an inexact Newton-type scheme with iterated sensitivities (INIS). Unlike standard inexact Newton methods, this technique is shown to preserve the local contraction properties of the forward scheme based on a specific Jacobian approximation for the corresponding equality constraints. More specifically, local convergence for the Newton-type method on the forward problem is shown to be necessary, and under mild conditions even sufficient for the asymptotic contraction of the corresponding INIS-type optimization algorithm. Together with the previous chapter on lifted collocation integrators, we presented how the INIS algorithm can be implemented efficiently, resulting in a computational cost close to that of the standard inexact Newton implementation. In addition, an adjoint-free (AF-INIS) variant is proposed

and its local convergence properties are studied. This alternative approach can be preferable whenever the algorithm can be carried out independent of the current values for the multipliers corresponding to the equality constraints.

The application and numerical benchmarking of the INIS optimization algorithm for different classes of dynamic optimization problems, e.g. those including partial differential equations (PDE), is part of ongoing research.

Chapter 8

Open-Source ACADO Code Generation Software

Following the active development of tailored optimization algorithms, many software packages are currently available for direct optimal control. For example, `MUSCOD-II` [92] is a multistage dynamic optimization software based on direct multiple shooting and SQP [209]. The software `dsoa` [104] is an optimal control tool based on single shooting. In addition to these shooting-based software packages, there are other approaches based on direct collocation which typically combine an AD tool [141] with a general-purpose sparse NLP solver such as `Ipopt` [318]. A few examples of such software packages are `CasADi` [20], `GPOPS-II` [247] and `PROPT` [303]. An important contribution of this thesis is the open-source implementation of the presented algorithmic techniques in the `ACADO Toolkit` [176] for nonlinear optimal control, as a part of its code generation tool, which has been presented in [177, 271]. In the context of real-time optimal control on embedded hardware, the technique of automatic code generation experienced an increasing popularity over the past decade [222, 237]. Alternative software packages for real-time NMPC are, for example, `OptCon` [298], `NEWCON` [285] and `VIATOC` [185].

Outline The chapter is organized as follows. Section 8.1 introduces the `ACADO` code generation software as part of the `ACADO Toolkit`. Section 8.2 briefly discusses interesting real-world control and estimation applications, and Section 8.3 presents the `ACADO` generated integrators with tailored sensitivity propagation. Section 8.4 finally concludes this chapter and provides a brief outlook on the future of optimal control software.

8.1 ACADO Code Generation Tool

`ACADO Toolkit` is a software environment and algorithm collection for automatic control and dynamic optimization. It is an open-source project for which the development originally started at KU Leuven, written in C++, and released under the GNU Lesser General Public License (LGPL). It provides a general framework for using a great variety of algorithms for direct optimal control, including MPC as well as state and parameter estimation. It also provides efficiently implemented RK and BDF integrators for the simulation of ODE and DAE systems, with a corresponding sensitivity analysis feature. The `ACADO Toolkit` has been designed with the following four key properties in mind: open-source, user-friendliness, extensible and self-contained code [176]. The following problem classes are mainly supported:

- Offline dynamic optimization problems
- Multi-objective optimization and OCPs
- Parameter and state estimation problems
- Online algorithms for nonlinear MPC and MHE

The rather intuitive syntax of the `ACADO Toolkit` allows the user to formulate control problems in a way that is very close to the usual mathematical syntax. The software can be used either directly from C++ or from its `MATLAB` interface. It can be downloaded freely from [3] and user support is typically provided through the active discussion forum [5].

8.1.1 Automatic Code Generation

In addition to using tailored algorithms, highly efficient code implementations are necessary to meet the tight timing constraints on embedded control hardware for which one additionally might need to use a specific, typically less powerful, programming language. In this context, the technique of automatic code generation has experienced an increasing popularity for real-time optimal control [222, 237]. One can perform offline optimizations of the code to be generated, e.g., by removing unnecessary computations, by optimizing memory access and cache usage and by exploiting the problem's specific structure. The software `CVXGEN` [221] and `FORCES` [97] are, for example, both generating customized IP solvers for convex optimization. Figure 8.1 illustrates the general code generation principle. The idea is to obtain a custom solver based on a

high-level problem description. This solver can then be used to efficiently solve specific instances of this problem formulation.

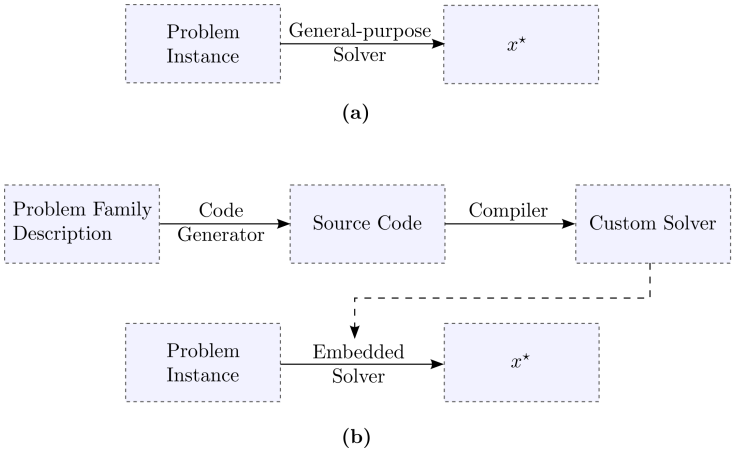


Figure 8.1: Illustration of (a) a general-purpose solver versus (b) the principle of code generation to obtain a custom solver for solving instances of a certain problem formulation (inspired by [222]).

More than 20 years ago, a code generation environment was already presented in [238] to export tailored implementations of Karmarkar's algorithm for solving Linear Programs (LP). About one decade ago, the software `AutoGenU` was developed to provide code generation for nonlinear MPC solutions [235, 237]. More recently, code generation has attracted great attention due to the software package `CVXGEN` [221], which can be tested by academic users via the web interface. The Multi-Parametric Toolbox (`MPT`) also provides automatic code generation for explicit MPC solutions [202]. Other examples of code generation packages based on online optimization tools are `µAO-MPC` [335] and `FiOrdOs` [306] for linear MPC or the new version of `AutoGenU` from `Maple` [236] and `VIATOC` [185] for nonlinear MPC.

The main motivation to use code generation is because it can speed-up the computations by removing unnecessary operations as well as by optimization of the generated source code. The latter consists of an optimized memory management because problem dimensions and sparsity patterns can be detected and hard-coded, a more efficient cache usage by reorganizing the computations and other techniques like loop unrolling. The second reason is the increased adaptivity of the numerical algorithm that will be exported. When implementing code generation, it becomes easier to offer options to, e.g., choose between single

and double precision arithmetic, avoid the use of certain libraries that are unavailable on the target hardware or to even choose the programming language in which the optimization code is exported [108].

These motivations to use code generation are clearly more relevant to real-time optimization of fast systems such as they appear in robotics and mechatronics, because of the rather high sampling rates. In addition, real-time optimization algorithms are often run on embedded hardware imposing extra restrictions which can be taken into account when exporting the code.

8.1.2 Open-Source ACADO Code Generation Tool

The `ACADO` code generation tool is a specific part of the `ACADO Toolkit`, which can be used to obtain real-time feasible codes for dynamic optimization on embedded control hardware. In particular, it directly pursues the export of highly efficient, self-contained C-code based on the RTI scheme for Nonlinear MPC (NMPC) [177]. `ACADO` code generation has already been used for real-time optimal control, showcasing milli- or even microsecond execution times both in simulation [10, 116, 148, 311] and in real-world experiments [9, 76, 129, 198, 309, 315]. As illustrated in Figure 8.2, a user friendly `MATLAB` interface is available which allows one to export, compile and use auto generated code in an intuitive way and without direct interaction with C/C++ programming [271]. Our open-source software implementation is mostly self-contained except for relying on tailored QP solvers for solving the optimal control structured subproblems [192]. In addition to AD techniques [176] and efficient integration schemes with sensitivity propagation [271], the use of a tailored convex solver is important to obtain a high performance for the overall SQP method.

More specifically, the open-source convex solvers `qpOASES` [109], `qpDUNES` [117], `FORCES` [97] and `HPMPC` [123] have been interfaced for direct optimal control. It is important to stress that these code generation techniques are not merely restricted to problems with a short control horizon and a small dynamic system. As discussed in [192, 314], a smart choice of the approach to solve the QP subproblem allows one to efficiently treat short to long horizon control problems. A statement on the system dimensions that can possibly be handled by the `ACADO` code generation framework would however strongly depend on the targeted application. Extensive benchmarking results of nonlinear MPC algorithms based on the `ACADO` code generation tool in combination with different convex solvers, can be found in [313]. In addition, multiple real-world applications are described further as well as in Chapter 9.

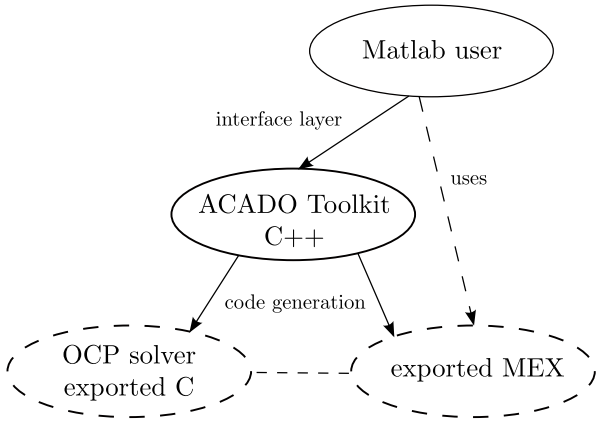


Figure 8.2: Illustration of the layers in the typical workflow when using the `MATLAB` interface of the `ACADO` code generation tool.

8.1.3 Nonlinear MPC using ACADO Code Generation

To obtain a real-time feasible implementation of NMPC, one can export a tailored RTI scheme using the open-source `ACADO` code generation tool. Figure 8.3 shows, for example, the `MATLAB` code structure as presented also in [271] for the optimal swing-up of an inverted pendulum mounted on top of a cart. Note that the nonlinear OCP formulation in Eq. (1.6) is supported even though a least squares stage cost is typically defined as in (1.24) or (1.25) for tracking MPC, such that users can explicitly specify weighting matrices and a reference trajectory. But also alternative objective functions are possible, as discussed in [268] for economic MPC based on `ACADO` code generation. The following five crucial steps can be identified in order to generate an NMPC solver:

1. Define the model equations: this part of the code in Figure 8.3 depends mostly on the specific optimal control application. The nonlinear system dynamics can be described directly in `ACADO` symbolics such that one can rely on the implemented AD routines or it can be provided based on external C functions, in which case also its first and possibly higher order derivatives are needed. The model can be defined in different ways, ranging from an explicit ODE to a fully implicit index-1 DAE formulation.

2. SIM export: as discussed further in Section 8.3, a stand-alone integrator can be exported using `ACADO` code generation. This feature can be used, e.g., to numerically simulate the system dynamics in a closed-loop NMPC

simulation. Alternatively, an application specific simulation environment or even an experimental setup could be used for this purpose.

3. Formulate the optimal control problem: the specific OCP needs to be formulated in `ACADO` syntax, including the system dynamics, the objective function and constraints. For this purpose, nonlinear functions can again be described either in `ACADO` symbolics or as external C-routines.

4. NMPC export: based on the given problem formulation, a corresponding online solver is exported using `OCPexport`. Various options are available to further customize the code generated algorithm. One can design the parameterization of the continuous time OCP and choose between different algorithmic components, such as integrators and convex solvers.

5. Closed-loop simulation: the exported solver can be used to run NMPC simulations for various closed-loop scenarios, including the efficiency of the optimized C-code in order to accurately assess real-time feasibility based on the corresponding computational effort.

Figure 8.3 illustrates the above steps for the NMPC example of the pendulum swing-up. Specific keywords are used to define the various optimization variables such as differential states and control inputs. They can be used to build all symbolic expressions in the `ACADO Toolkit`. After defining the model, the code exports the 4-stage ERK method of order 4 as a stand-alone integrator to simulate the system. It is specified to perform 5 integration steps to simulate the model over 0.05 s, resulting in a relatively high accuracy. The `SIMexport` module generates the optimized C-code into the folder called 'SIM_export'. Notice that there are options to customize the integrator such as to include an efficient propagation of certain sensitivities. This turns it into a powerful tool to prototype new algorithms as discussed in Section 8.3. The third and fourth step involve the use of the `OCP` and `OCPexport` module, which generates the RTI based optimal control algorithm for NMPC. It allows the user to specify the embedded components such as the integrator and the convex solver. In the code from Figure 8.3, the same ERK method is used and combined together with a condensing technique and `qpOASES` as the underlying solver. As mentioned earlier, it is possible to alternatively employ a tailored structure exploiting solver such as `qpDUNES` or `HPMPC`. The self-contained C-code is exported into the specified 'MPC_export' folder in this case.

The `ACADO` code generation tool provides the user with easy access to the exported code to perform various NMPC simulations from `MATLAB`. The presented code in Figure 8.3 automatically generates MEX functions which can be used to call the exported integrator and RTI based solver directly from `MATLAB`, in order to, e.g., perform a closed-loop simulation. Alternatively, `ACADO`

```
%% ---  1) Define the model equations  ---
DifferentialState p theta dp dtheta     % define states
Control u                               % define controls

l = 0.5;  g = 9.81;  m = 0.1;  M = 1;
expr1 = -m*cos(theta)^2 + M + m;
expr2 = m*l*sin(theta)*dtheta^2 + u;
expr3 = m*g*sin(theta);
ode = [ dot(p); dot(theta); dot(dp); dot(dtheta) ] == ...
    [ dp; dtheta; (expr2 + expr3*cos(theta))/expr1; ...
     -(cos(theta)*expr2 + expr3 + M*g*sin(theta))/(l*expr1) ];

%% ---  2) Export integrator  ---
sim = acado.SIMexport( 0.05 );  % sampling time Ts = 0.05s
sim.setModel( ode );
sim.set( 'INTEGRATOR_TYPE', 'INT_RK4' );  % integration method
sim.set( 'NUM_INTEGRATOR_STEPS', 5 );     % number of steps
sim.exportCode( 'SIM_export' );  % export code and compile
make_acado_integrator( '../acado_system' )

%% ---  3) Formulate OCP  ---
ocp = acado.OCP( 0.0,1.0,20 );  % 20 control intervals over 1s
W = diag([10 10 0.1 0.1 0.01]); % least squares stage cost
ocp.minimizeLSQ( W, [p; theta; dp; dtheta; u] );
ocp.minimizeLSQEndTerm( W(1:4,1:4), [p; theta; dp; dtheta] );
ocp.subjectTo( -2.0 <= p <= 2.0 );   % state bounds
ocp.subjectTo( -20.0 <= u <= 20.0 ); % control bounds
ocp.setModel( ode ); % constraints from dynamic model

%% ---  4) Export NMPC solver  ---
mpc = acado.OCPexport( ocp );
mpc.set( 'QP_SOLVER', 'QP_QPOASES' );
mpc.set( 'INTEGRATOR_TYPE', 'INT_RK4' );
mpc.set( 'NUM_INTEGRATOR_STEPS', 20 );  % steps over horizon
mpc.exportCode( 'MPC_export' );  % export code and compile
make_acado_solver( '../acado_RTIstep' )

%% ---  5) Closed-loop simulation  ---
time = 0;  Ts = 0.05;  Tf = 5;
input.y = repmat([0 pi 0 0 0], 20, 1);  % reference trajectory
input.yN = [0 pi 0 0];
input.x = zeros(21, 4);  % initial state and control values
input.u = zeros(20, 1);
state_sim = zeros(4,1);
while time < Tf
    input.x0 = state_sim.';  % current state values
    output = acado_RTIstep(input);  % one RTI step
    input.x = [output.x(2:end,:); output.x(end,:)]; % shifting
    input.u = [output.u(2:end,:); output.u(end,:)];

    input_sim.x = state_sim;
    input_sim.u = output.u(1,:).';
    output_sim = acado_system(input_sim);  % simulate system
    state_sim = output_sim.value;
    time = time + Ts;
end
```

Figure 8.3: MATLAB code example to implement NMPC using ACADO code generation for the optimal swing-up of an inverted pendulum on top of a cart.

can also automatically generate a MATLAB S-function for the solver in order to perform the closed-loop simulation from Simulink. Note that the same optimized C-code could later be employed in real-time on embedded control hardware, which is often achieved by using the Simulink Coder. The open-source software can be downloaded from www.acadotoolkit.org to reproduce any of the numerical results presented within this thesis.

8.2 Real-Time Control Applications

Let us briefly discuss some of the many realized applications of the ACADO code generation tool for real-time optimal control and refer to the corresponding publications. Chapter 9 later presents the specific case of Nonlinear MPC for a two-stage turbocharged gasoline engine in more detail.

8.2.1 Overhead Crane Setup

A schematic description of the considered overhead crane is given in Figure 8.4. The pendulum consists of a cylindrical load hanging on two parallel cables. A cart can position the load in the x-direction while a winching mechanism can position the load in the y-direction. Reliable sensors are available on the cart position x_{C}, cable length x_{L} and cable angle θ. The system inputs are the voltages u_{C} and u_{L}, representing setpoints for the respective internal velocity controller. Note that the overhead crane optimal control setup was already used in Section 4.3.1 to illustrate the performance of the structure exploiting integrators within nonlinear MPC. The dynamic model as described in [76, 316], consists of only a few nonlinearly defined states while the remaining differential equations are linear. A detailed description of the system can be found in [313]. The control software is implemented using the OROCOS Toolchain [58] and runs on a PC with an Intel Xeon 2.53 GHz quad core processor, 12GB RAM memory, and a preemptive Linux kernel as operating system. The sampling frequency of the estimation and control is fixed to 100 Hz. Experimental results have been presented respectively in [316] for nonlinear MPC and in [76] for the combined MHE-NMPC approach. The computationally most efficient practical implementation for this optimal control application was based on the embedded collocation integrators from Chapter 2, to deal with the stiff dynamics, and using the structure exploitation as described in Chapter 4.

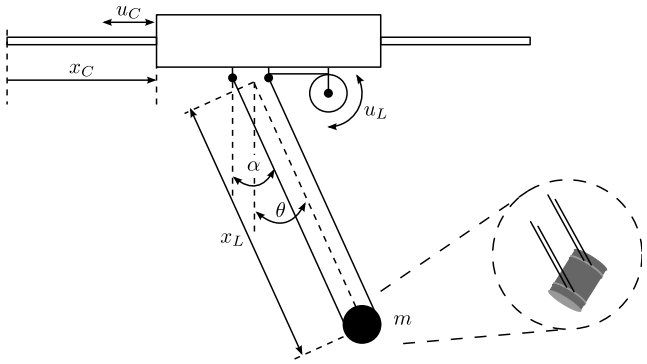


Figure 8.4: Schematic of the overhead crane experimental setup [316].

8.2.2 Airborne Wind Energy

The idea of Airborne Wind Energy (AWE) aims at harvesting energy by using a kite in crosswind flight, i.e., in the direction perpendicular to the air flow. As introduced originally in [217], this AWE concept forms an alternative approach to classical wind turbines which allows one to fly at much higher altitudes with higher wind speeds while considerably reducing the corresponding structural costs. The system can produce energy by either using on-board generators or by pulling the tether while flying a periodic power generating orbit [85]. The latter ground-based form of power generation is often referred to as *pumping*, versus an on-board generation or *drag* mode. In addition to being a very nonlinear and unstable system to control, this specific application also provides many algorithmic challenges. The multi-body AWE models [329] need to account for the winch, the tether and the aerodynamics of the airfoil, which typically results in complex index-3 DAE models based on Lagrange mechanics. For this purpose, a carousel setup has been built at KU Leuven within the framework of the ERC Highwind project in order to test the hardware, software and online algorithms within a controlled environment. This particular system was designed [127] in order to investigate a rotational startup and landing procedure as discussed in more detail in [329]. Experimental results for the deployment of MHE and NMPC based on `ACADO` code generation, including the embedded simulation methods for index-1 DAEs as discussed in Chapter 2, can be found in [129, 313, 315] for this setup. The work in [331] on the real-time control of dual-airfoil systems, for example, presents total computation times below 125 ms for a control horizon of 20 intervals and using a nonlinear DAE model of 56 differential, 3 algebraic states and 14 control inputs.

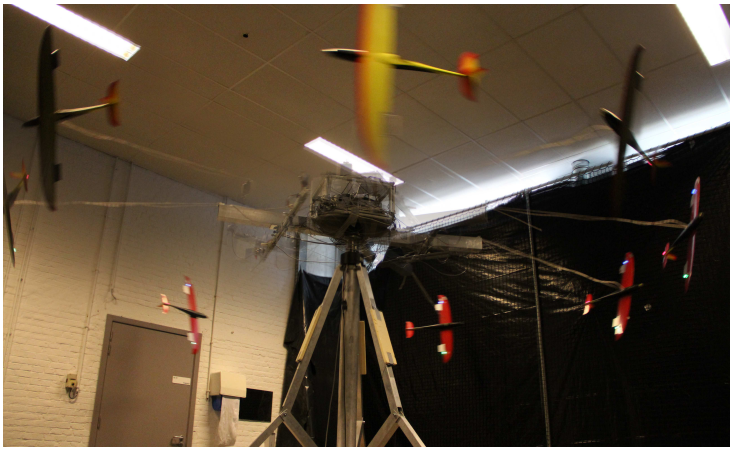


Figure 8.5: Illustration of the kite carousel setup in motion at KU Leuven [313].

8.2.3 Diesel Engine Airpath Control

Predictive control based on online optimization techniques forms a promising approach for diesel engine control, considering the incrementally more stringent emission legislation in combination with the increasing level of system complexity and the corresponding limitations. We consider a passenger car 2l 4 cylinder turbocharged diesel engine, equipped with a variable geometry turbocharger and an external exhaust gas recirculation system. Figure 8.6 presents a sketch of the resulting diesel engine airpath system with its main components. This work builds on the results presented in [110] of `qpOASES` based linear MPC for real-world diesel engine control on embedded hardware, with sampling times in the order of milliseconds. A data-driven polynomial Nonlinear AutoRegressive model with eXogenous (NARX) input has been identified, in order to obtain a multiple-input multiple-output (MIMO) description of the airpath system. For this purpose, tailored support for NARX based NMPC has been implemented in the `ACADO` code generation tool as can be seen further in Figure 8.9. Real-time control experiments have been performed with an `ACADO` generated NMPC solver on the testbench at the Johannes Kepler University (JKU) in Linz, and this using a dSpace Autobox 1006 with an AMD processor from `MATLAB Simulink`. A detailed publication on NMPC based diesel engine control is currently under preparation, including these experimental results.

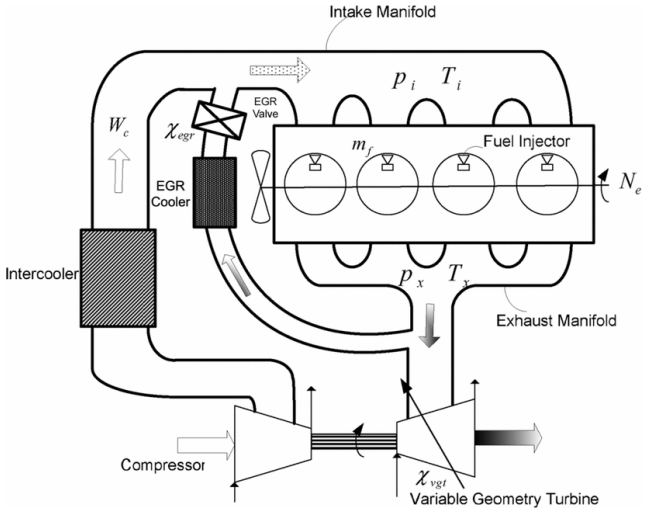


Figure 8.6: Sketch of a diesel engine airpath with its main components [41].

8.2.4 Time-Optimal Race Car Driving

Fully autonomous driving typically requires more advanced control paradigms, such as the MPC techniques in [105, 125]. An `ACADO` generated real-time NMPC implementation was presented in [116], based on a detailed vehicle model consisting of 14 states and including a Pacejka tire model. Time-optimal driving provides additional challenges, since the controller needs to act fast while coping with the nonlinear vehicle dynamics and satisfying the track boundaries. In order to validate the proposed methods, an experimental setup has been built with small-scale race cars at the university of Freiburg. Note however that a similar setup was built earlier at LMS, Siemens in Leuven [309], following the experimental setup at ETH in Zurich [213]. Measurements on the vehicle's current position, orientation, velocity and yaw rate are obtained through a custom-made camera system with a sampling speed of 100 Hz. The actuation signal is sent from the real-time Linux station to the car via a wireless connection. The considered race track features a chicane, a U-turn and a longer straight section. Further technical details on the experimental setup can be found in [308], as illustrated also in Figure 8.7. A near time-optimal NMPC implementation was presented in [309], including experimental results. More recently, the use of an exact Hessian based RTI scheme for time optimal racing based on an economic MPC formulation has been presented in [311], using the symmetric Hessian propagation as proposed in Chapter 3.

Figure 8.7: The miniature race car setup at the University of Freiburg.

8.3 ACADO Integrator Code Generation

Figure 8.8 illustrates the two main modules in the `ACADO` code generation tool in the form of a simplified class inheritance diagram. The `OCPexport` module has been discussed mainly in the previous sections in order to automatically generate real-time feasible optimal control solvers, e.g., to implement online MHE or NMPC. In addition, the `SIMexport` functionality provides the option to generate a stand-alone, embeddable integrator code with tailored sensitivity propagation. Both modules inherit from the class `ExportModule` in Figure 8.8, which represents the export of any set of auto generated, tailored algorithms. The `SIMexport` module was originally developed for the purpose of testing the integrator codes with sensitivity analysis [259], while it became a powerful tool for rapid prototyping of interesting nonlinear optimal control algorithms [271]. Examples of the latter can be found as part of the DMS based RTI scheme of Chapter 5, the block based ALADIN algorithm in [194], the Sequential Semidefinite Programming (SSDP) method for optimal experiment design in [300] or the continuous output based MHE in [193, 261].

Both modules can rely on the code generation classes in Figure 8.9 to export specific algorithmic components, such as integrators, linear solvers, Hessian approximation and condensing schemes. They use other more internal code generation classes for the export of data variables, arithmetic statements, functions and interfaces for which we omit the corresponding C++ class hierarchies but we refer the interested reader to the documentation of the open-source `ACADO Toolkit` software. The condensing options, for example, include the classical $\mathcal{O}(N^3)$ condensing [51], the alternative $\mathcal{O}(N^2)$ algorithm [19],

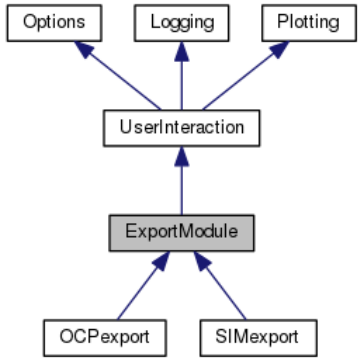


Figure 8.8: Illustration of the two modules in the ACADO code generation tool.

block condensing [192] and the scheme providing a factorized condensed Hessian [122]. More information on when to use which method can be found in [121]. The exported RTI algorithm can be either based on a Gauss-Newton method or it uses an exact Hessian scheme as discussed in [268]. Note that the ACADO code generation tool supports the export of self-contained and problem specific C-code for linear algebra routines, such as the matrix factorization which is for example crucial for the efficient implementation of implicit integration schemes. The algorithms include a QR factorization using Householder triangularization, an LU factorization based on Gaussian elimination and a Cholesky decomposition [137]. In addition, tailored features have been implemented such as multiple backsolves, to perform transposed system solutions and complex arithmetics for the simplified Newton iterations as described in Chapter 2.

Figure 8.9 also illustrates the extensive inheritance diagram starting from the `IntegratorExport` class. The work throughout this thesis has focused on the use of RK integration schemes, represented by the `ExplicitRungeKuttaExport` and `ImplicitRungeKuttaExport` code generation classes. Following the object-oriented programming paradigm, it is relatively easy to include new formulas based on their Butcher tableau [157, 158]. Currently, the ERK methods available in the code generation tool are the following:

- `ExplicitEulerExport`: the explicit Euler method of order 1
- `ExplicitRungeKutta2Export`: the ERK method of order 2 (midpoint)
- `ExplicitRungeKutta3Export`: the ERK method of order 3 (Heun)
- `ExplicitRungeKutta4Export`: the ERK method of order 4 (RK4)

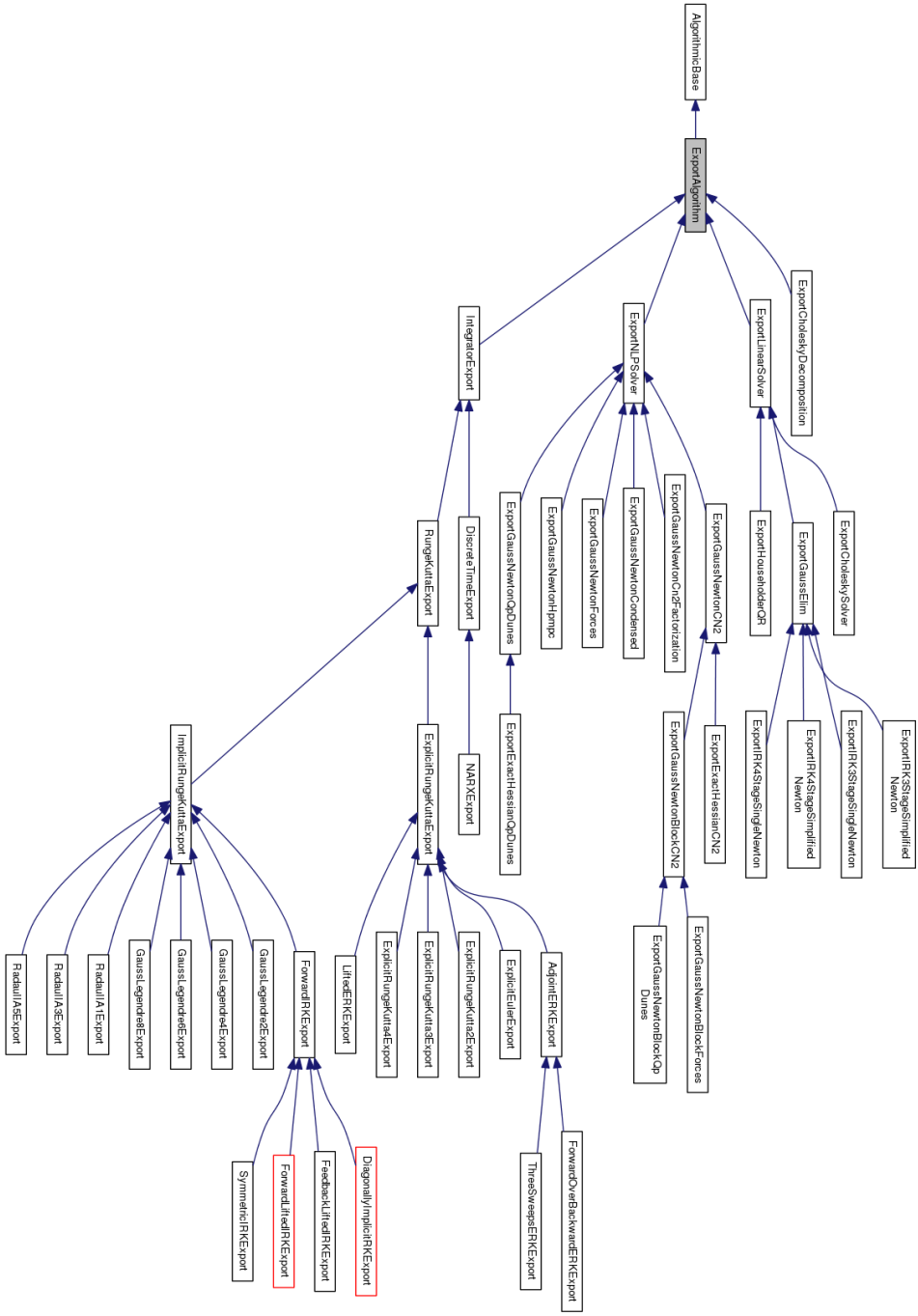


Figure 8.9: Overview of the main ACADO code generation classes to export tailored algorithms.

The fully implicit RK methods in the code generation tool are:

- `RadauIIA1Export`: Radau IIA method of order 1
- `GaussLegendre2Export`: Gauss method of order 2
- `RadauIIA3Export`: Radau IIA method of order 3
- `GaussLegendre4Export`: Gauss method of order 4
- `RadauIIA5Export`: Radau IIA method of order 5
- `GaussLegendre6Export`: Gauss method of order 6
- `GaussLegendre8Export`: Gauss method of order 8

The above collocation methods include the continuous output feature as described earlier in Chapter 2. In addition, the tool supports the export of the Diagonal IRK (DIRK) methods of order 3, 4 and 5 in [158].

The majority of the above integration schemes have been extended with both forward and backward sensitivity propagation techniques. Also, the classical forward-over-adjoint and the novel symmetric Hessian propagation schemes from Chapter 3 have been implemented for both the explicit and implicit methods. Note that sensitivity analysis of arbitrary order is possible, for example, in `DAESOL-II` [6] or as part of `CasADi` [19] based on advanced AD techniques. Finally, the IRK integrators can be used with the IFT-R scheme, an exact Newton method or any of the exact and inexact adjoint-based or INIS lifting variants as presented in Chapter 6. Extensive benchmarking results of the stand-alone RK integrators with tailored sensitivity propagation using the `ACADO` code generation tool can be found in [259, 265, 269, 271]. Even though [269] showed that the `ACADO` auto generated methods can be approximately 100 times faster than the solvers from `SUNDIALS` and a speedup factor of about 500 was observed in [261] with respect to `ode15s`, it should be noted that these implementations have rather different purposes. The integrators in the `SUNDIALS` package [166], for example, are based on a general-purpose implementation of variable-stepsize variable-order methods for the numerical simulation and sensitivity analysis of large-scale systems. On the other hand, the `ACADO` integrator suite is targeted at very fast, real-time and embedded applications for small to medium-scale ODE or index-1 DAE systems.

8.4 Conclusions and Outlook

One contribution of this thesis is that the proposed algorithms have been implemented as part of the open-source ACADO code generation tool, resulting in an increased visibility and applicability of the presented work. It is however difficult to avoid that parts of big open-source projects can become unmaintained or outdated relatively quickly. This is especially the case when the software development is performed within academic research, where the number of active developers can strongly vary over periods of multiple years. This said, the same open-source software can continue to be useful for many researchers. The package can, for example, aid to initiate new software development projects or can be used to develop and prototype new algorithmic techniques. For this reason, to increase the chance of code reuse and to facilitate rapid prototyping of algorithms, there has been a strong need for *modularity* in optimal control software as mentioned earlier in this chapter. One would ideally have access to the different algorithmic components such as AD, numerical integration with sensitivity propagation, convex and non-convex solvers etc., from a common high-level modeling environment. In the case of real-time optimal control, one additionally needs the option to transform the resulting algorithm into an embeddable solver in a relatively easy manner.

Following these conclusions, a new optimal control software project has been under development, based on the ACADO code generation tool, which should become much easier to maintain and of which the code can be reused in a more straightforward way. It consists of separate modules for each identifiable component, with each their own clearly defined interface such that they can be combined in different ways to create tailored optimization algorithms. The core components are written directly in an embeddable fashion based on a standard C-code implementation, such that the need for an intermediate code generation layer can be reduced mainly to a customization functionality. One can rely on hardware specific code optimizations in the main linear algebra routines to obtain the desired numerical performance, as discussed in more detail in [121, 123]. Based on the success of the ACADO integrator suite and the proposed algorithmic techniques, an extended and more self-contained implementation can be envisioned in a similar fashion and this for both small, medium or even large-scale dynamic systems.

Chapter 9

Two-Stage Turbocharged Gasoline Engine

Based on the proposed algorithmic techniques and the open-source software developments within the ACADO code generation tool, let us focus in this chapter on a particular real-world case study. The investigated application is the control of a complex airpath structure, which consists of a two-stage turbocharging concept for gasoline engines. The goal is to realize fast reference tracking for the boost pressure without overshoot or offset, while respecting constraints on the turbocharger speeds to prevent damages. For this purpose, NMPC will be shown to be a rather promising approach that achieves a high control quality while respecting the limitations of the system. The chapter includes a detailed description of the system set-up and the corresponding optimal control problem formulation. Closed-loop numerical simulations on the dSpace MicroAutoBox are used to motivate the real-time feasibility and control performance of the NMPC scheme, followed by the real-world experimental results based on the implementation within a demonstrator vehicle.

Note that this presentation is largely based on the article in [12], which itself is built upon earlier publications as part of multiple conference proceedings. The reduced-order state space model has been proposed and validated based on closed-loop numerical simulations in [10], while the real-time feasibility and tuning of the resulting NMPC scheme has been studied in [9]. The presented work has been carried out as part of a research collaboration between the University of Freiburg and the RWTH Aachen University. Especially Thivaharan Albin and Dennis Ritter, the main authors of the aforementioned publications, have greatly contributed to this work, including the experimental results.

Outline The chapter is organized as follows. Section 9.1 provides a general but compact introduction to airpath control, followed by an overview on the particular two-stage turbocharging concept for gasoline engines in Section 9.2. The reduced-order modeling of the system is described in Section 9.3 and the implementation of the corresponding NMPC scheme is discussed in Section 9.4. Section 9.5 presents the closed-loop simulation results for the control algorithm. The validation of the controller using experiments on a vehicle dynamometer is described in Section 9.6, which additionally presents the vehicle experiments on the road while driving dynamically.

9.1 Introduction to Airpath Control

To reduce fuel consumption and emissions for internal combustion engines, *downsizing* by the use of turbochargers is investigated. For increasing the specific power, conventional single-stage turbocharging concepts lead to conflicting goals concerning the dimensioning of the charging components. A high specific power on the one hand and a fast transient raise of the boost pressure on the other hand cannot be realized at the same time. For mitigation of this trade-off, more variability in the charging devices is used.

A future promising technology, in particular for gasoline engines, is the *two-stage turbocharging* concept. The architecture consists of a small high-pressure (HP) stage and a large low-pressure (LP) stage. The small high-pressure stage is capable of realizing fast transients, even though it is restricted concerning the specific power. In contrast to that, the large low-pressure stage can realize a high specific power with slower transient dynamics. One of the arising challenges consists in the design of the closed-loop controller which fulfills the high requirements on the control quality.

9.1.1 Real-Time Control Requirements

The control algorithm needs to handle both turbocharger stages in a coordinated fashion, such that reference tracking for the boost pressure is made possible and disturbances are rejected quickly. Note that for gasoline engines, the boost pressure correlates directly to the driving torque. The reason lies in the quantitative control used as working principle for gasoline engines, which results in the need for a fixed air-to-fuel ratio and thus the torque is determined by the airpath system. The boost pressure reference should therefore be reached as quickly as possible, as this determines the transient acceleration capability of the vehicle. In the case of a step reference input, the output should additionally

be achieved without strong overshoots as this influences the driving behaviour negatively. In diesel engines, oscillations can be tolerated up to a certain amount since the driving torque is determined by the fuel path. This is due to the qualitative control used as working principle for diesel engines, which allows for variations in the air-to-fuel ratio which therefore decouples the torque and boost pressure to a certain extent.

In addition to these requirements on the reference tracking performance, the control algorithm should respect the constraints for the high- and low-pressure turbocharger speeds, since exceeding these limits might damage the turbocharger. This becomes especially challenging, as the turbocharger speed is typically not even measured in a series-production configuration. In summary, the following three major requirements have to be fulfilled at the same time for gasoline airpath control:

- fast boost pressure reference tracking,
- without strong oscillations,
- while respecting limits on turbocharger speeds.

These control requirements are quite demanding, e.g., compared to diesel engine airpath control. Note that turbocharged systems are strongly nonlinear and their dynamic behaviour is very dependent on disturbance variables such as the engine speed. For this reason, the design of an NMPC scheme is investigated in this chapter, which is a suitable choice as it can handle the nonlinear system dynamics with high control quality and is able to respect constraints on system states. The overall goal is that the NMPC scheme makes use of the specific turbocharging architecture to overcome the trade-off between fast transient raise and high power.

9.1.2 Literature Review on Airpath Control

An overview on fundamental modeling and control of airpath systems can be found in [103] and [153]. For the purpose of airpath control, a variety of concepts have been applied where especially model-based control shows advantages due to the arising strong nonlinearities. Examples of the latter are internal model control [258] and flatness-based control [191]. Compared to other model-based control concepts, the use of MPC shows advantages as it is able to inherently consider constraints on the actuated signals as well as on the system states [11]. In the case of airpath control, it is an important feature to not only consider the limitations of the actuators but, for example, also the constraints on the turbocharger speeds.

Among the applied MPC concepts, online optimization has been investigated as well as offline optimization based implementations. Explicit MPC is used, e.g., in [100] based on a piecewise-affine (PWA) approximation of the system behaviour. On the other hand, also different MPC concepts based on online optimization have been developed where different measures were used to take the nonlinearity into account. The majority of the current publications use linear time varying (LTV) MPC, which means that the optimal control problem is based on a linearized model gained at the recent operating point, see [11] or [84]. More advanced nonlinear MPC control techniques were recently investigated in [164], [179] and [229], where it is applied to single-stage diesel engines with a data-based dynamic model.

However, for the control of the more complex two-stage gasoline turbocharging architecture, only a relatively small amount of publications is available. The work in [136] presents a vehicle set-up for which the authors utilize a single-input single-output (SISO) proportional–integral–derivative (PID) controller. For rather low engine speeds, the high-pressure stage is controlled (low-pressure wastegate fully open) and for higher engine speeds, the low-pressure stage is controlled (high-pressure wastegate fully open). In the medium speed range, one of the two actuated values is applied in a feedforward-manner with look-up tables, the other input is based on feedback control. In [301], the modeling and control of the pneumatic actuation system for a 2-stage gasoline airpath architecture is discussed. An MPC approach based on a PWA approximation of the turbocharging concept has been presented in [11]. The PWA based MPC scheme works well for small load steps, e.g., steps starting at throttled operation to a boost pressure of 1.4 bar. For steps to boost pressures larger than 1.5 bar, where the nonlinearity is stronger, the control results show considerable overshoots which are not tolerable for real-world driving.

9.2 Two-Stage Turbocharged Gasoline Engine

For notational convenience, we further use the variables and indices as they have been introduced in the corresponding publications in [9, 10, 12].

9.2.1 Airpath System Architecture

Figure 9.1 shows the schematic set-up of the investigated airpath architecture. For experimental analysis, the depicted system has been built up and implemented in a demonstrator vehicle (Ford Focus) with a 1.8 l 4-cylinder gasoline engine as illustrated in Figure 9.2. The vehicle allows for real-world

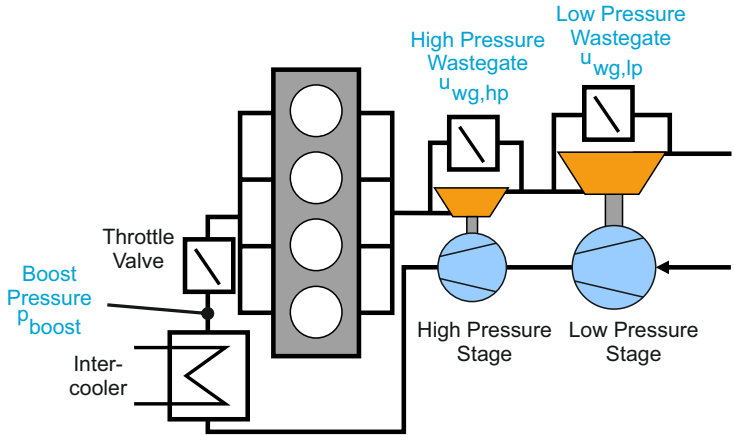


Figure 9.1: System overview of the investigated two-stage turbocharging concept

Figure 9.2: Illustration of the demonstrator vehicle and the testing track.

experiments to validate the control performance. The algorithm has been implemented on a rapid prototyping hardware, i.e., the dSpace MicroAutoBox. The closed-loop system has been tested using experiments on a vehicle dynamometer as well as by performing driving experiments on the road. A more detailed overview on the turbocharging system is given in [60].

9.2.2 Available Sensors

The controlled variable of the system is the boost pressure p_{boost}, which is measured with a pressure sensor positioned behind the intercooler and in front of the throttle valve. Due to the high exhaust gas temperatures in gasoline engines, the application of sensors in the exhaust gas path is problematic for a series configuration due to durability and price issues. Thus, in the presented approach, no sensor signal from the exhaust gas path is used for control purposes.

Additionally measured variables are the engine speed n_{eng} and the ambient pressure p_{amb}. All these sensors that are used for control, are available in a series-production configuration. For the purpose of modeling and validation of the algorithm, additional sensors are implemented in the car. For example, both turbocharger speeds and the pressure between the compressors (resulting in the pressure ratio over the two compressors) are measured in the demonstrator vehicle but only for the modeling and validation process.

9.2.3 Actuation System

As actuators for the control of the turbochargers, wastegates on the high-pressure ($u_{\mathrm{wg,hp}}$) and on the low-pressure stage ($u_{\mathrm{wg,lp}}$) are used. Electronic wastegates are commonly used for turbocharging, which have the advantage that they have a position feedback-sensor and thus allow for accurate setting of the valve opening area. In this work, the use of simpler and cheaper pneumatic actuators is investigated. They do not use any additional sensor, e.g., for position feedback, which makes the control more demanding. The wastegate actuation signals correspond to a pulse-width modulated (PWM) signal, which has an allowable operating range $0 \leq u_{\mathrm{wg},\star} \leq 100$ for $\star = \mathrm{lp}, \mathrm{hp}$. It manipulates the pilot pressure, which has influence on the cross-section diameter of the opening. Thus, the amount of exhaust gas that passes the turbine or the wastegate can be adjusted. Low values such as $u_{\mathrm{wg},\star} = 0$ open the wastegate and higher values such as $u_{\mathrm{wg},\star} = 100$ close the wastegate. In the case of a fully open wastegate, the majority of the exhaust gas bypasses the turbine. Whereas in the case of a fully closed wastegate, all the exhaust gas flows through the turbine. In addition, a throttle valve is present in the considered architecture which is controlled via a separate function, not part of the turbocharger control. Note that at very high mass flows, the high-pressure stage does not deliver boost pressure anymore. However, with the given sizing of the components, the high-pressure stage does not go into a safety critical mode here. This is why no additional high-pressure bypass for the compressor is used in the control concept, accepting some efficiency losses at high mass flows.

9.2.4 Engine Control Algorithm

In summary, the controlled variable is the boost pressure p_{boost} and the actuated values are the high-pressure and low-pressure wastegate PWM signals $u_{\mathrm{wg,hp}}$ and $u_{\mathrm{wg,lp}}$. Additionally, the engine speed n_{eng} and ambient pressure p_{amb} are used as measured disturbances. The sampling time for the airpath control has been chosen to be $t_{\mathrm{samp}} = 25$ ms. As is nowadays usual in series applications,

the throttle valve is only used in the non-boosted region. Within the boosted region, the throttle valve is set completely open for reasons of fuel efficiency. As a consequence, the boost pressure is equal to the intake manifold pressure and directly correlated to the torque of the engine. Therefore, the throttle valve is not investigated further. The setpoint for boost pressure is determined by the requested torque in a conventional manner, as given in torque oriented engine control structures. All other parameters of the engine control structure, such as ignition, injection and camshaft position are based on the standard calibration. The airpath NMPC control algorithm is implemented on dSpace MicroAutoBox. All other engine functions, such as setpoint calculation, ignition timing, etc., are implemented on the same control hardware.

9.3 Modeling of the Airpath System

In the following, the nonlinear state space model is introduced which is used within the NMPC problem formulation. For the controller internal model, the focus is set on capturing the system dynamics while trying to keep the function outputs as well as their derivatives smooth and the state dimensions small. This is generally important as one of the hurdles for NMPC is typically the computational cost during operation. A nonlinear state space model will be presented, which is mainly driven by physical equations. This dynamic model is shown to reproduce the measurement data quantitatively well in a big operating range concerning engine speed and load.

9.3.1 Fundamental Turbocharging Equations

Well-investigated physical equations are used as basis for the model, whereas a detailed overview on turbocharger modeling can be found in [153]. The power on the turbine $\Psi_{\mathrm{tur},\star}$ and compressor $\Psi_{\mathrm{com},\star}$ are used to define the power balance on the high- and low-pressure stage as described in Eq. (9.1a). The expressions in (9.1b)-(9.1c) relate the power on the two stages of the compressor and respectively the turbine to the mass flow through the corresponding device and the total change of enthalpy. In these equations, the pressure ratio is given by $\Pi_{\mathrm{tur}} = \frac{p_{\mathrm{utur}}}{p_{\mathrm{dtur}}}$ and $\Pi_{\mathrm{com}} = \frac{p_{\mathrm{dcom}}}{p_{\mathrm{ucom}}}$. The expressions with $\star = \mathrm{lp}, \mathrm{hp}$ hold for

both the high- and low-pressure stage;

$$\Psi_{\mathrm{tur},\star} - \Psi_{\mathrm{com},\star} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{2}\Theta_{\mathrm{turbo},\star}\, n_{\mathrm{turbo},\star}^2\right) \tag{9.1a}$$

$$\Psi_{\mathrm{com},\star} = \dot{m}_{\mathrm{com},\star}\, c_{p,\mathrm{air}}\, T_{\mathrm{ucom},\star} \frac{1}{\eta_{\mathrm{is,com},\star}} \left(\Pi_{\mathrm{com},\star}^{\frac{\kappa_{\mathrm{air}}-1}{\kappa_{\mathrm{air}}}} - 1\right) \tag{9.1b}$$

$$\Psi_{\mathrm{tur},\star} = \dot{m}_{\mathrm{tur},\star}\, c_{p,\mathrm{exh}}\, T_{\mathrm{utur},\star}\, \eta_{\mathrm{is,tur},\star} \left(1 - \Pi_{\mathrm{tur},\star}^{\frac{1-\kappa_{\mathrm{exh}}}{\kappa_{\mathrm{exh}}}}\right). \tag{9.1c}$$

The different arising mass flows, such as the aspirated and the fuel mass flows, are calculated based on the models from [227]. To obtain the mass flow through the high- and low-pressure wastegate and through the turbine, the throttle equation can be used as described in [10, 12].

9.3.2 Reduced-Order Modeling

Various simplifications have been carried out in order to gain a dynamic model which is suitable for real-time NMPC, as described in the following.

Rotational kinetic energy and pressure ratio

A common approach to simplify the model, is to correlate the rotational kinetic energy to the pressure ratio over its corresponding compressor. In the case of diesel engines, for example, this can be done by linear maps [227]. For gasoline two-stage turbocharging, a distinction has to be made between the high- and low-pressure stage. For the low-pressure stage, an affine map with the parameters a_{lp} and b_{lp} can be fitted based on measurement data:

$$n_{\mathrm{turbo,lp}}^2 = a_{\mathrm{lp}} \Pi_{\mathrm{com,lp}} + b_{\mathrm{lp}}. \tag{9.2}$$

On the high-pressure stage, a large spread of exhaust enthalpy and compressor operating points is present such that the dependency on the engine speed should additionally be taken into account. The following bilinear relation has shown to work well in practice [10, 12]:

$$n_{\mathrm{turbo,hp}}^2 = (a_{\mathrm{hp}} n_{\mathrm{eng}} + c_{\mathrm{hp}})\, \Pi_{\mathrm{com,hp}} + (b_{\mathrm{hp}} n_{\mathrm{eng}} + d_{\mathrm{hp}})\,. \tag{9.3}$$

Singular perturbation theory

For the gasoline two-stage turbocharging, the model simplification by singular perturbation theory is applicable as described in [187]. This results in the following equations:

$$\dot{m}_{\mathrm{com,lp}} = \dot{m}_{\mathrm{com,hp}} = \dot{m}_{\mathrm{asp}} \tag{9.4}$$

and

$$\begin{aligned} \dot{m}_{\mathrm{asp}} + \dot{m}_{\mathrm{fuel}} &= \dot{m}_{\mathrm{tur,hp}} + \dot{m}_{\mathrm{wg,hp}} \\ &= \dot{m}_{\mathrm{tur,lp}} + \dot{m}_{\mathrm{wg,lp}}. \end{aligned} \tag{9.5}$$

System input delay

Based on the available vehicle implementation of the airpath architecture, a considerable dead time is observed in the system. This effect can be described, e.g., based on additional models of the pipings and the pneumatic actuation system, but this would lead to quite complex overall dynamics. Instead, the dead time is assumed to be a constant input delay arising from the actuation signal. Based on measurement data, this dead time has been estimated to be $t_{\mathrm{D}} = 0.45$ s, which makes the control task more challenging.

Modeling of the wastegate opening area

In the given set-up, pneumatic systems are used for actuating the wastegate opening area. The high-pressure wastegate is actuated via underpressure and the low-pressure stage via excesspressure, which makes the behaviour of each stage relatively different. The actuation signal $u_{\mathrm{wg,hp}}$ adjusts the ratio of underpressure delivered by a vacuum pump, which is mixed with ambient pressure. On the other hand, the low-pressure stage $u_{\mathrm{wg,lp}}$ uses the boost pressure to control the wastegate. In both cases, the opening area can be calculated based on the corresponding force equilibrium. Compared to the airpath system, the dynamics of the actuator are quite fast such that it can be treated as a static relationship. For gaining cheap computations, the opening area characteristic has been approximated by a smooth function. The high-pressure stage opening area $A_{\mathrm{wg,hp}}$ correlates linearly with the actuated signal. For the low-pressure stage, the dependency of the opening area $A_{\mathrm{wg,lp}}$ on the current boost pressure cannot be neglected [10, 12].

9.3.3 Resulting State Space Model

The resulting DAE system will be used as the state space model within the NMPC scheme. The dynamic system is governed by introducing the differential states $x_1 = \Pi_{\text{com,lp}}$, $x_2 = \Pi_{\text{com,hp}}$, and the algebraic variables $z_1 = \Pi_{\text{tur,lp}}$, $z_2 = \Pi_{\text{tur,hp}}$. The model constants can be summarized as $c_1, \ldots, c_8$, which are defined as part of the detailed discussion in [12]. Based on these parameters and setting $\kappa_{\text{air}} = 1.4$ as well as $\kappa_{\text{exh}} = 1.33$, the resulting DAE system reads as:

$$\begin{aligned}
\dot{x}_1(t) &= c_1\, p_{\text{amb}} \left(z_1^{1.5} - z_1^{1.25}\right) \sqrt{z_1^{-1.5} - z_1^{-1.75}} \\
&\quad - c_2\, p_{\text{amb}}\, n_{\text{eng}}\, x_2 \left(x_1^{1.29} - x_1\right) && \text{(9.6a)} \\
\dot{x}_2(t) &= c_5\, p_{\text{amb}}\, z_1 \left(z_2^{1.5} - z_2^{1.25}\right) \sqrt{z_2^{-1.5} - z_2^{-1.75}} \\
&\quad - c_6\, p_{\text{amb}}\, n_{\text{eng}}\, x_1 \left(x_2^{1.29} - x_2\right) && \text{(9.6b)} \\
0 &= x_1\, x_2 - c_3 \frac{1}{n_{\text{eng}}} \sqrt{z_1^{0.5} - z_1^{0.25}} \left(\sqrt{z_1} + c_4\, A_{\text{wg,lp}}(x_1 x_2, u_{\text{wg,lp}})\right) && \text{(9.6c)} \\
0 &= x_1\, x_2 - c_7 \frac{1}{n_{\text{eng}}} z_1 \sqrt{z_2^{0.5} - z_2^{0.25}} \left(\sqrt{z_2} + c_8\, A_{\text{wg,hp}}(u_{\text{wg,hp}})\right), && \text{(9.6d)}
\end{aligned}$$

with the corresponding output functions:

$$\begin{aligned}
y_1 &= p_{\text{amb}}\, x_1\, x_2 \\
y_2 &= \sqrt{(a_{\text{lp}} x_1 + b_{\text{lp}})} \\
y_3 &= \sqrt{(a_{\text{hp}} n_{\text{eng}} + c_{\text{hp}})\, x_2 + (b_{\text{hp}} n_{\text{eng}} + d_{\text{hp}})}.
\end{aligned} \tag{9.7}$$

The DAE model is of index 1 (see Definition 1.3) and consists of 2 differential and 2 algebraic states, the output y_1 corresponds to the boost pressure p_{boost} which is reference tracked. The extra outputs y_2 and y_3 correspond to the low-$n_{\text{turbo,lp}}$ and the high-pressure $n_{\text{turbo,hp}}$ turbocharger speeds, where the goal is to constrain them within the optimal control problem.

9.3.4 Reduced-Order Model Validation

In order to validate the proposed model, measurement data was gained using the demonstrator vehicle for constant as well as for varying engine speeds.

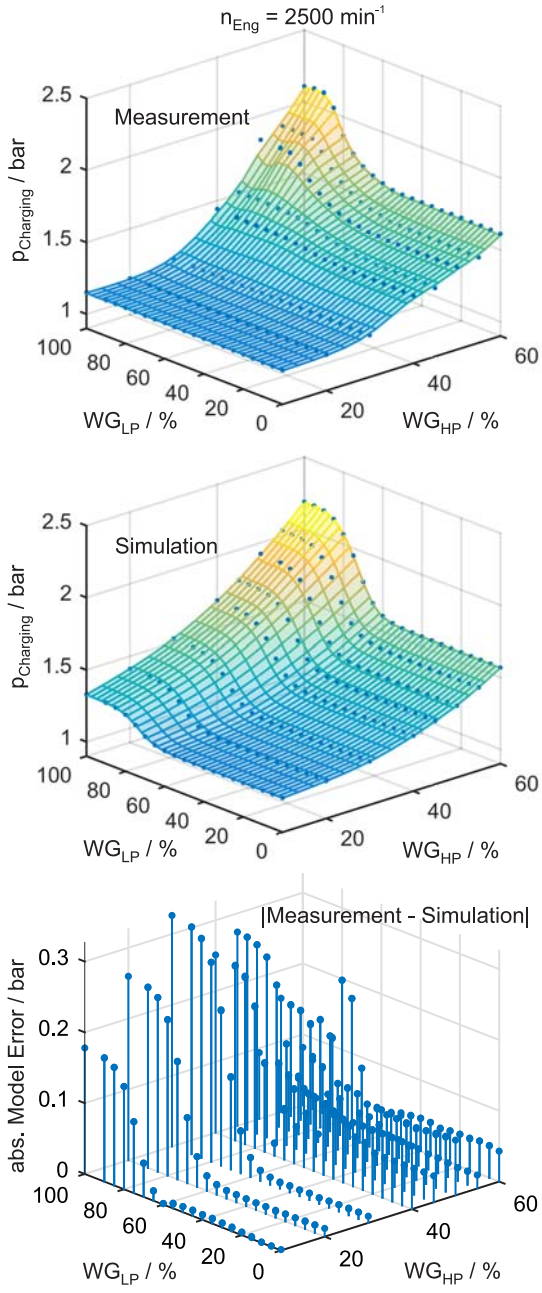


Figure 9.3: Validation of the stationary transfer behaviour of the DAE model, based on corresponding measurement data at $n_{\text{eng}} = 2500$ min^{-1}.

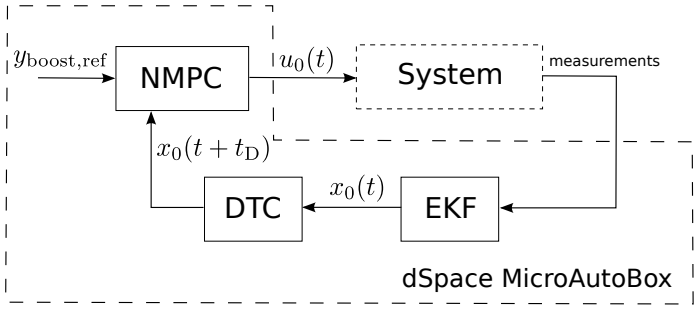


Figure 9.4: Illustration of the closed-loop system on the dSpace MicroAutoBox: NMPC based on `ACADO` code generation with dead time compensator (DTC) and extended Kalman filter (EKF).

For measurements at a constant speed, a dynamometer test bench has been used where the engine speed was set at different values between $n_{\text{eng}} = 1500$ min^{-1} and $n_{\text{eng}} = 3500$ min^{-1}. Figure 9.3 shows the result of such a validation experiment. The upper plot shows the steady state measurement data for the boost pressure p_{boost}, recorded on a grid of different values for $u_{\text{wg,lp}}$ and $u_{\text{wg,hp}}$ at a constant engine speed of $n_{\text{eng}} = 2500$ min^{-1}. The lower plot shows the steady state data for the boost pressure p_{boost}, obtained by simulating the dynamic model in (9.6). From this comparison, one can see that the model qualitatively and quantitatively reproduces the measurement data very well. The same holds for the dynamic data, e.g., using a step response, as well as for the comparison at different engine speeds.

9.4 Nonlinear MPC and State Estimation

This section gives an overview on the entire closed-loop system, as illustrated in Figure 9.4. The key element is the NMPC scheme based on the embedded optimization and simulation methods within the `ACADO` code generation tool as presented earlier. An extended Kalman filter (EKF) is used to observe the state of the system and additionally a dead time compensator (DTC) is implemented to account for the considerable dead time.

9.4.1 Disturbance Model and State Observer

One of the goals for the closed-loop controller is to achieve offset-free tracking even in the presence of disturbances and model mismatch. More precisely, the

goal is that the boost pressure y_{boost} is reference tracked without offset for the case that the reference and the disturbances are asymptotically constant. For this purpose, the expression of the boost pressure is adapted such that $y_{\text{boost}} = p_{\text{amb}}\, x_1\, x_2 + d$ with the disturbance dynamics $\dot{d}(t) = 0$. By augmenting the state vector $\tilde{x}(t)^\top = \begin{bmatrix} x(t)^\top & d(t) \end{bmatrix}$, the overall system equations including the disturbance model can be summarized as:

$$\begin{aligned} 0 &= f(\dot{\tilde{x}}(t), \tilde{x}(t), z(t), u(t - t_{\text{D}})) \\ y &= \psi(\tilde{x}(t)). \end{aligned} \tag{9.8}$$

An EKF is used for the estimation of the system state and disturbances. It updates the estimates with a sampling time $t_{\text{samp}} = 0.025$ s. For the relation between continuous and discrete time, it follows that $t_k = t_0 + k\, t_{\text{samp}}$ and the dead time $k_{\text{D}} = t_{\text{D}}/t_{\text{samp}}$. Denoting $\tilde{x}^-_{\text{es},k}$ as the a priori state estimate at time index k, the a priori error covariance as P_k^- as well as $J_{x,k} = \frac{\partial \phi}{\partial x_k}(\tilde{x}_{\text{es},k}, u_{k-k_{\text{D}}})$ and $J_{y,k} = \frac{\partial \psi}{\partial x_k}(\tilde{x}^-_{\text{es},k})$, the prediction step reads as

$$\begin{aligned} \tilde{x}^-_{\text{es},k} &= \phi(\tilde{x}_{\text{es},k-1}, u_{k-1-k_{\text{D}}}) \\ P_k^- &= J_{x,k-1}\, P_{k-1}\, J_{x,k-1}^T + Q_{\text{KF}}. \end{aligned} \tag{9.9}$$

The function $\phi(x_i, u_i)$ denotes the numerical simulation of the nonlinear dynamics throughout one sampling step, starting from the given state x_i and using the control inputs u_i. Since the model consists of an index-1 DAE system in Eq. (9.6), an implicit integration method with corresponding sensitivity analysis is used as discussed in Chapter 2.

In the subsequent correction step, the a priori estimates are updated based on the recent measurements. The resulting values of this calculation are the a posteriori estimates for the system state $\tilde{x}_{\text{es},k}$ and the error covariance P_k. The following equations result for this correction step:

$$\begin{aligned} L_k &= \frac{P_k^-\, J_{y,k}^T}{J_{y,k}\, P_k^-\, J_{y,k}^T + R_{\text{KF}}} \\ \tilde{x}_{\text{es},k} &= \tilde{x}^-_{\text{es},k} + L_k\, (y_{\text{meas},k} - \psi(\tilde{x}^-_{\text{es},k})) \\ P_k &= (\mathbb{1} - L_k\, J_{y,k})\, P_k^-. \end{aligned} \tag{9.10}$$

9.4.2 Dead Time Compensation

The EKF provides the current state estimate $\tilde{x}_{\text{es},k}$, which can be used along with the model (9.8) in the NMPC algorithm to compute the next control

input. However, the dead time can be treated with an alternative approach which simplifies the optimization task. For this purpose, a delay-free model is considered based on the same system dynamics:

$$\begin{aligned} 0 &= f(\dot{\hat{x}}(t), \hat{x}(t), z(t), u(t)) \\ y &= \psi(\hat{x}(t)). \end{aligned} \tag{9.11}$$

The new state variable $\hat{x}_k$ corresponds to the state $\tilde{x}_{k+k_\mathrm{D}}$ which is shifted by the dead time, and the NMPC scheme uses this delay-free model instead. The predicted state $\hat{x}_{\mathrm{es},k}$ needs to be obtained, using the inputs that have already been applied to the plant. Note that for this purpose, the last k_D inputs need to be stored. Using $\Phi(\tilde{x}_i, u_{i-k_\mathrm{D}}, \ldots, u_{i-1})$ to denote the numerical integration from time point i to $i + k_\mathrm{D}$, starting from the initial state $\tilde{x}_i$ with the actuated signals $u_{i-k_\mathrm{D}}, \ldots, u_{i-1}$, it follows that:

$$\hat{x}_{\mathrm{es},k} = \Phi(\tilde{x}_{\mathrm{es},k}, u_{k-k_\mathrm{D}}, \ldots, u_{k-1}). \tag{9.12}$$

9.4.3 Optimal Control Problem Formulation

The NMPC scheme needs to solve one nonlinear OCP at each sampling instant. For simplicity of notation, we further use $x(t)$ directly to refer to the state variable of the delay-free model in Eq. (9.11). Let us introduce the following continuous time OCP formulation:

$$\min_{x(\cdot),\, u(\cdot)} \quad \int_0^T \ell\left(x(t), u(t)\right) \mathrm{d}t + m\left(x(T)\right) \tag{9.13a}$$

$$\text{s.t.} \quad 0 = x(0) - \hat{x}_{\mathrm{es}}, \tag{9.13b}$$

$$0 = f(\dot{x}(t), x(t), z(t), u(t)), \qquad \forall t \in [0, T], \tag{9.13c}$$

$$\underline{y}_2 - s(t) \leq y_2(t) \leq \bar{y}_2 + s(t), \qquad \forall t \in [0, T], \tag{9.13d}$$

$$\underline{y}_3 - s(t) \leq y_3(t) \leq \bar{y}_3 + s(t), \qquad \forall t \in [0, T], \tag{9.13e}$$

$$\underline{u}_{\mathrm{wg,lp}} \leq u_{\mathrm{wg,lp}}(t) \leq \bar{u}_{\mathrm{wg,lp}}, \qquad \forall t \in [0, T], \tag{9.13f}$$

$$\underline{u}_{\mathrm{wg,hp}} \leq u_{\mathrm{wg,hp}}(t) \leq \bar{u}_{\mathrm{wg,hp}}, \qquad \forall t \in [0, T], \tag{9.13g}$$

$$\underline{\dot{u}}_{\mathrm{wg,lp}} \leq \dot{u}_{\mathrm{wg,lp}}(t) \leq \bar{\dot{u}}_{\mathrm{wg,lp}}, \qquad \forall t \in [0, T], \tag{9.13h}$$

$$\underline{\dot{u}}_{\mathrm{wg,hp}} \leq \dot{u}_{\mathrm{wg,hp}}(t) \leq \bar{\dot{u}}_{\mathrm{wg,hp}}, \qquad \forall t \in [0, T], \tag{9.13i}$$

$$0 \leq s(t), \qquad \forall t \in [0, T], \tag{9.13j}$$

where the stage and terminal cost functions are defined as

$$\ell\left(x(t),u(t)\right) = \|y_1(t) - y_{\mathrm{boost,ref}}(t)\|_Q^2 + \|u(t)\|_R^2, \tag{9.14a}$$

$$m\left(x(T)\right) = \|y_1(T) - y_{\mathrm{boost,ref}}(T)\|_{Q_N}^2. \tag{9.14b}$$

This nonlinear OCP depends on the parameter $\hat{x}_{\mathrm{es}} \in \mathbb{R}^{n_{\mathrm{x}}}$, which denotes the current state estimate, through the initial value condition of Eq. (9.13b). The slack variable $s(t)$ is used to define the soft constraints in (9.13d)-(9.13e), which allows for preferably small violations of the original output bounds. The change rates of the control inputs and the slack variable are optimized directly $u(t) := [\dot{u}_{\mathrm{wg,lp}}(t), \dot{u}_{\mathrm{wg,hp}}(t), s(t)]^\top \in \mathbb{R}^3$, such that the wastegate actuation signals are formulated as part of the system state vector $x(t) := [\Pi_{\mathrm{com,lp}}(t), \Pi_{\mathrm{com,hp}}(t), u_{\mathrm{wg,lp}}(t), u_{\mathrm{wg,hp}}(t)]^\top \in \mathbb{R}^4$ and the algebraic variables $z(t) := [\Pi_{\mathrm{tur,lp}}(t), \Pi_{\mathrm{tur,hp}}(t)]^\top \in \mathbb{R}^2$ are defined. The implicit dynamics in (9.13c) are given by the DAE system in Eq. (9.6), in addition to the input dynamics, which is of index 1 such that the Jacobian matrix $\frac{\partial f(\cdot)}{\partial(z,\dot{x})}$ is invertible as in Definition 1.3. The objective consists of a least squares type tracking cost as defined by Eqs. (9.14a)-(9.14b). Note that the reference signal $y_{\mathrm{boost,ref}}(t)$ is a constant value over the prediction horizon in this case. The path constraints consist of simple bounds defined by Eqs. (9.13d) to (9.13j).

9.4.4 Real-Time Optimal Control

This continuous time OCP formulation can be recognized to be of the rather standard form in Eq. (1.6). Direct optimal control methods and tailored algorithms can therefore be used to treat the embedded optimization problem, for which effective approaches have been proposed throughout this thesis. More specifically, direct multiple shooting for the OCP in (9.13) results in an NLP of the form in Eq. (1.8). The latter NLP does not include algebraic states in the decision variables, since they only enter the DAE system (9.13c) for this specific problem formulation. Note that the penalization of the control change rate is implemented by including an extra differential state which denotes the original control value, while its time derivative is defined as the new control input. Because of the least squares cost (9.13a), a Gauss-Newton based SQP algorithm can be used. To allow for a real-time feasible implementation, the RTI scheme will be used as introduced as part of Section 1.5 and this based on our implementation in the `ACADO` code generation tool (see Chapter 8). In order to solve the structured QP subproblem in each iteration, condensing will be used in combination with `qpOASES`.

9.5 Simulative Assessment of NMPC

Before testing the NMPC scheme in the vehicle, extensive closed-loop simulations have been conducted. The goal here is to decide on the parameterization of the online algorithm and the validation of the resulting scheme, concerning the aforementioned control requirements. In addition, the performance is compared to that of less advanced algorithms based on a linearized system model. The simulations were conducted on the control prototyping hardware, i.e., the dSpace MicroAutoBox, which is also used in the vehicle such that relevant computation times can be evaluated.

9.5.1 Parameterization of NMPC

In the case of NMPC, quite a few parameters have to be tuned to realize a closed-loop system that satisfies real-time feasibility while achieving a high control quality. An overview on some of the most important design parameters is given in Figure 9.5. The control horizon is set to $T = 1.5$ s, which allows one to sufficiently cover the dynamics until steady state of the system. If the horizon is chosen too short, the control performance decreases and in the worst case the turbocharger speed limits cannot be respected anymore. In order to maintain real-time feasibility, the NMPC scheme does not use $t_{\mathrm{samp}} = 25$ ms as a discretization step size. Instead, $N = 20$ equidistant shooting intervals are used which results in the discretization time $T_{\mathrm{s}} = 75$ ms. Extensive simulations have shown the algorithm to be more effective when using a larger discretization time along with a longer control horizon instead of using a more fine discretization of a shorter horizon [9]. An IRK scheme based on Gauss collocation of order 2 is used with a fixed integration step size of $T_{\mathrm{int}} = 75$ ms. To solve each QP subproblem within the RTI scheme, condensing with `qpOASES` is used with a maximum number of iterations $N_{\mathrm{QP}} = 50$ in order to constrain the runtime [9, 109]. Note that also other competitive approaches can be used as discussed earlier in Section 1.4. In addition, sufficiently large terminal weights are used for stability reasons [151] while mainly designing the objective to result in a fast reference tracking. For the EKF, the covariance matrices were chosen such that quick disturbance estimation is achieved as the present noise is rather weak.

9.5.2 Validation of NMPC

The functionality of the NMPC algorithm has been compared to that of less advanced MPC schemes in Figure 9.6. For this purpose, a linear time invariant (LTI) and a linear time varying (LTV) MPC were implemented. The

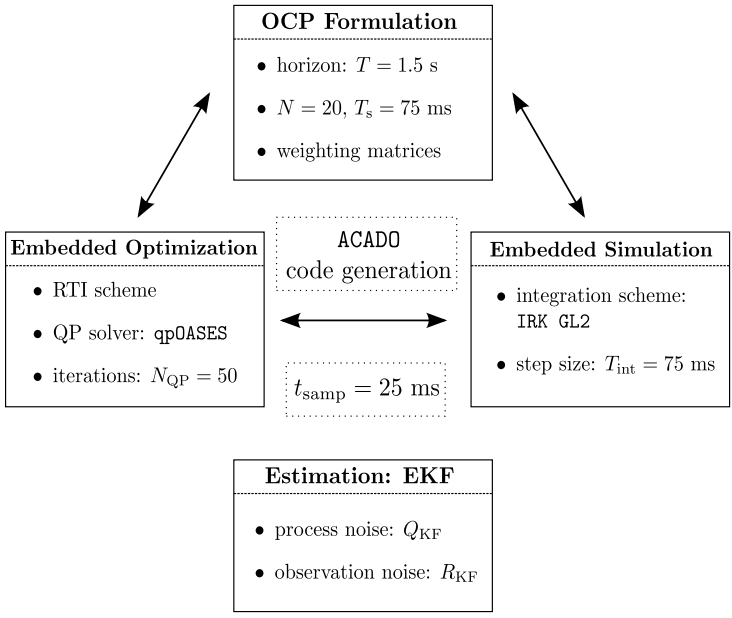


Figure 9.5: Overview on design parameters of the different components that are crucial for the real-time feasible NMPC scheme.

Table 9.1: Comparison of rise time t_{95} between nonlinear MPC (NMPC), linear time varying (LTV) MPC and linear MPC.

	Nonlinear MPC	LTV MPC	Linear MPC
Step to 2.2 bar	0.9 s	1.4 s	3.4 s
Step to 2.6 bar	1.1 s	1.5 s	3.5 s

LTI scheme uses one linear model over the entire operating region, whereas the LTV MPC calculates in every time step the linearized model around the recent operating point. In these two cases, the weighting matrices were chosen such that no exceeding of the turbocharger speed limits $n_{\text{turbo,hp}}$ and $n_{\text{turbo,lp}}$ arises and in order to obtain fast reference tracking without considerable overshoots over the entire operating range. The LTI, LTV and the NMPC scheme all result in stabilizing controllers, which are able to consider the constraints on the turbocharger speed limits and allow for offset-free reference tracking. However, a difference arises with respect to the value t_{95} which is the time needed for reaching 95% of the reference. Table 9.1 gives an overview on the rise time t_{95} for different load steps at nominal conditions for $n_{\text{eng}} = 1500$ min^{-1}.

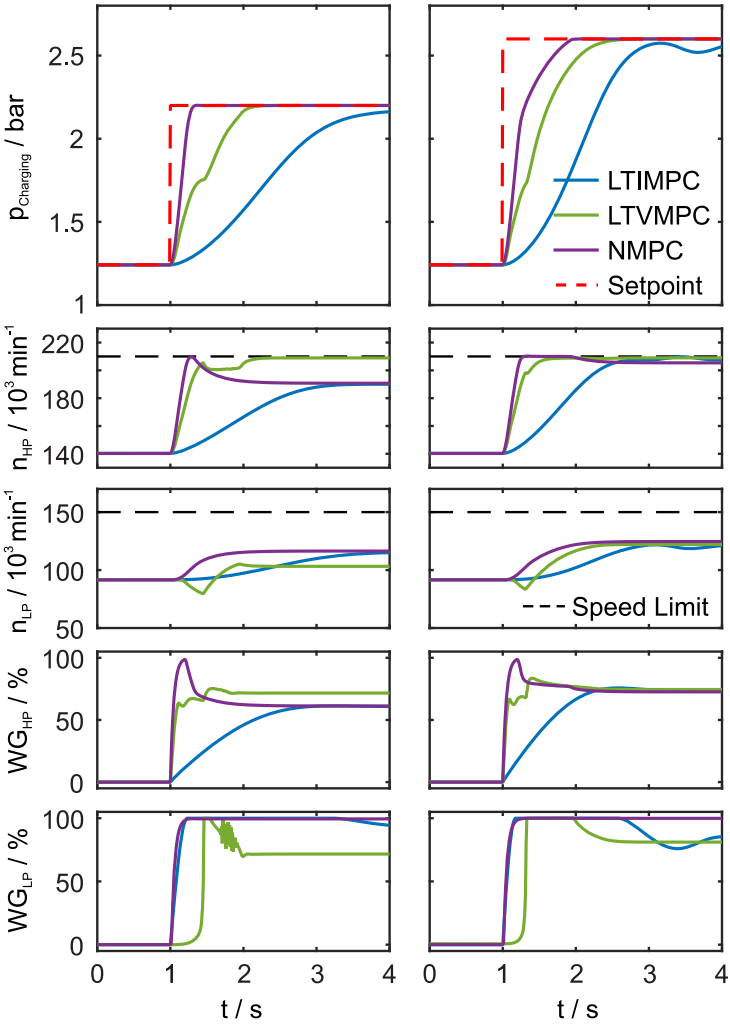


Figure 9.6: Closed-loop hardware-in-the-loop simulations: comparison of LTI, LTV and NMPC for a step in the boost pressure reference signal.

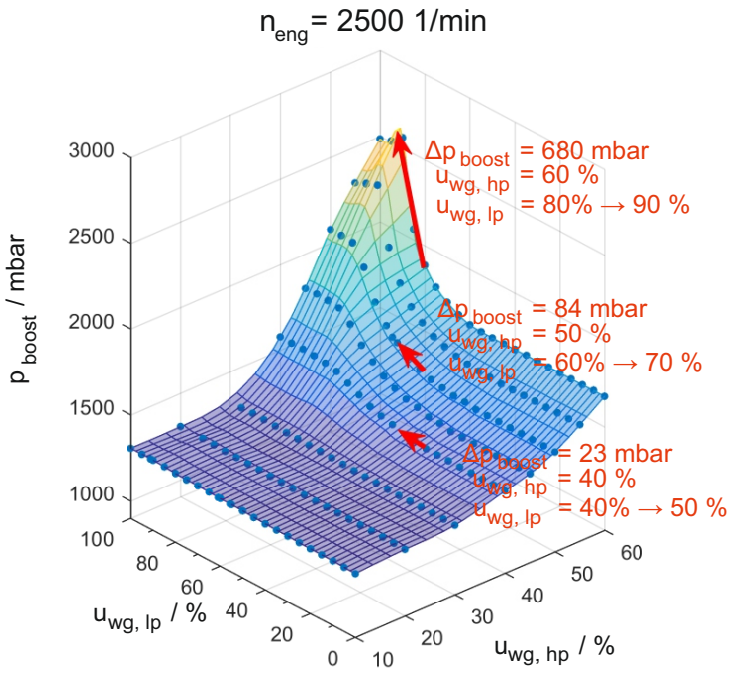


Figure 9.7: Steady state measurement data for $n_{\text{eng}} = 2500$ min^{-1}, with an illustration of the change in boost pressure for 10% wastegate actuation.

The linear MPC scheme shows a rather slow response behaviour. The difference between LTV and NMPC in the time t_{95} is around 0.5 s, which has a considerable and perceptible effect on performance. To explain this further, Figure 9.7 illustrates the steady state behaviour of the system with experimental data for a constant engine speed. Depending on the operating point, the gradients change significantly which has to be considered especially for a big step in boost pressure. In contrast to the other concepts, the NMPC is aware of these nonlinearities in advance such that the reference can be tracked much faster. If the control algorithm is not aware of the future nonlinear dynamics, the speed of reference tracking has to be slowed down in order to avoid overshoots which are not allowed in gasoline airpath control.

9.6 In-Vehicle Experimental Results

9.6.1 Validation on a Vehicle Dynamometer

The first step of experimental testing was done on a vehicle dynamometer. This offers the possibility to test the airpath control for the case of constant engine speed, which is a major disturbance variable. In Figure 9.8, one exemplary closed-loop control result can be seen. In the given case, a step is applied from the non-boosted region to $y_{\mathrm{boost,ref}} = 2.2$ bar at a constant engine speed of $n_{\mathrm{eng}} = 2500\ \mathrm{min}^{-1}$. The closed-loop control system is able to meet all specified criteria. It is possible to track the reference without overshoot and without offset, by rejecting all disturbances. At the same time, the controller is able to respect the high-pressure and low-pressure turbocharger speed limits. It can be observed that the control scheme is able to exploit the turbocharging architecture. First, the high-pressure stage is used to quickly increase boost pressure. However, it is only used as much such that the speed limit is not violated. Simultaneously, the speed of the low-pressure stage is increased. Both stages are balanced such that no overshoot is present, which is especially challenging as $y_{\mathrm{boost}} = y_{\mathrm{boost,ref}}$ is reached before the system goes to steady state. After reaching the setpoint, the actuation as well as the turbocharger speeds are still changing, which shows the advantage of the nonlinear predictive control.

In addition, Figure 9.9 shows the closed-loop results where the same boost pressure reference step is applied for different engine speeds. The control requirements are observed to be met for all engine speeds. Additionally, the pressure ratios are illustrated for each case in the same figure. They show that at all three engine speeds, the high-pressure stage is used for quick increase in boost pressure. At mid engine speeds in stationary operation, the control relies in a relatively similar fashion on the high- and on the low-pressure stage. At high engine speeds, the boost pressure is realized almost completely by the low-pressure stage. The control concept is able to account for the changeover between the two stages such that the design goals of the two-stage turbocharging are realized, i.e., based on fast dynamics with the high-pressure stage and high power with the low-pressure stage.

9.6.2 In-Vehicle Testing on the Road

The experiments on the vehicle dynamometer give important insights to the control system. As the process is however highly nonlinear, random reference values and disturbance signals can have a drastic impact on the control performance. For this reason, experimental testings on the road are inevitable

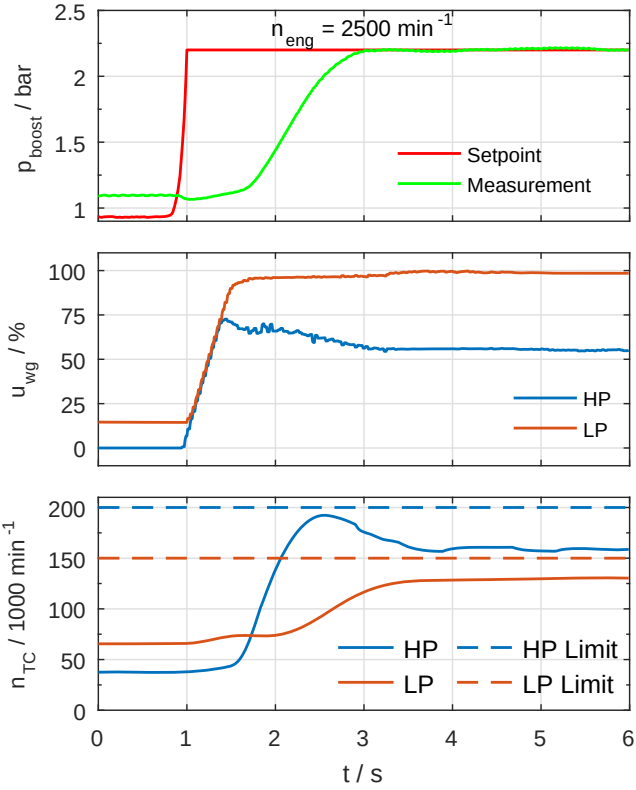


Figure 9.8: Vehicle dynamometer experiments: closed-loop control for a step in the boost pressure reference signal at engine speed $n_{\mathrm{eng}} = 2500$ min^{-1}.

to demonstrate functionality. For the road testing, random profiles have been driven on an automotive track. These real-world experiments have shown that the developed control algorithm is able to have a high closed-loop performance, in the entire operating range and for all drive profiles.

In Figure 9.10, an exemplary control result is shown where a vehicle acceleration was tested starting from 80 km/h to 120 km/h. One can see that the reference can be accurately tracked, despite the signal variations. A certain lag is present between the reference and the system output, which cannot be overcome as it results from the dead time of the system and the inertia of the turbochargers. For better evaluation, the boost pressure output profile is plotted once more but then shifted by the dead time of $t_{\mathrm{D}} = 0.45$ s. Just as in the case of the vehicle dynamometer testings, one can see that the control uses the high-pressure stage to quickly increase the boost pressure. It should be stressed that

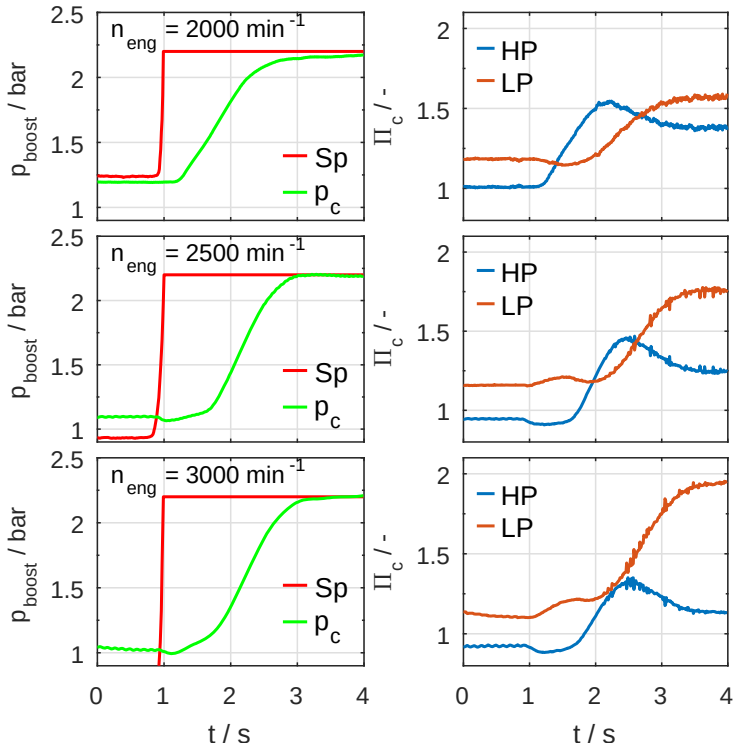


Figure 9.9: Vehicle dynamometer experiments: closed-loop control for a step in the boost pressure reference signal for different engine speeds.

the high-pressure stage is only used as much, so that the limit values on the turbocharger speed are taken into account, which intuitively corresponds to an optimal control strategy. Additionally, note that this performance is realized without using turbocharger speed sensors and only relying on the model (the plotted profile corresponds to measurement values, which were only used for validation purposes). The engaging and disengaging of the different stages works even for the case of speed transient operation.

9.7 Conclusions and Outlook

Novel airpath concepts have been investigated for gasoline engines, in order to reduce the fuel consumption. One promising approach is based on the serial two-stage turbocharging architecture. However, this airpath concept

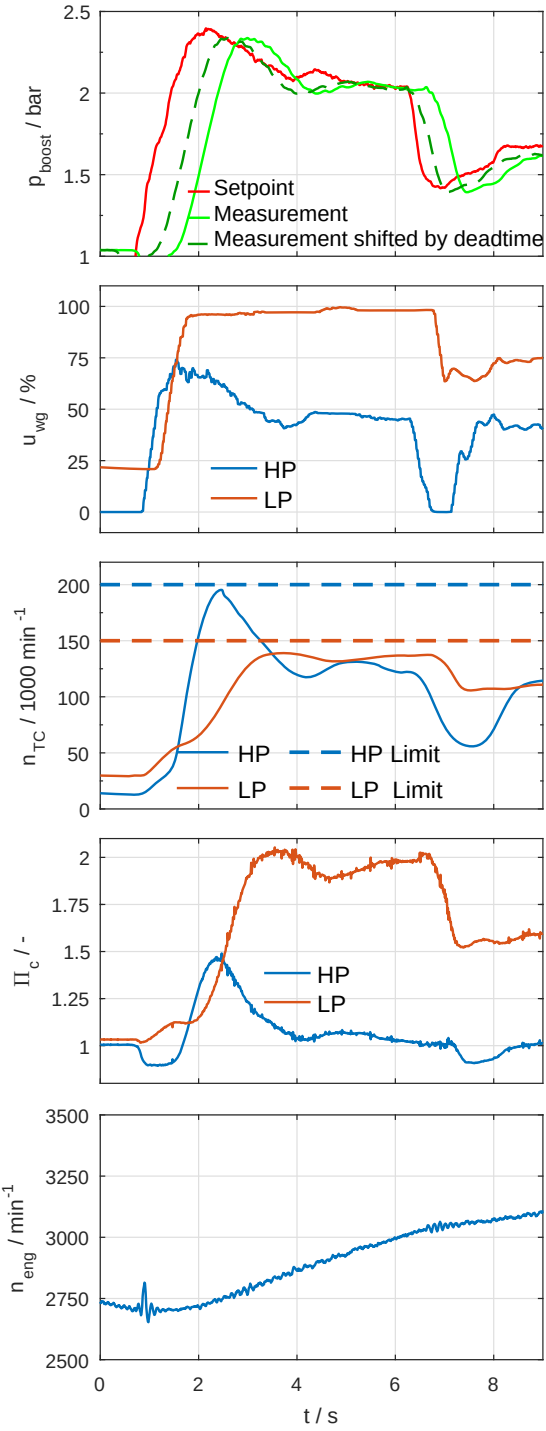


Figure 9.10: Closed-loop experimental results with the vehicle on the road.

exhibits strong nonlinear behaviour, while having high demands on the control quality and including strict system requirements. For this purpose, a real-time feasible NMPC scheme has been proposed, based on the RTI implementation in the open-source `ACADO` code generation tool. The basis of the NMPC scheme is a physically driven reduced-order state space model formulated as a set of differential-algebraic equations. Simulative testings have been performed, which show the advantage of NMPC over alternative approaches. Additionally, the algorithm has been evaluated in a vehicle where the airpath concept is implemented. These real-world experiments have been conducted on a vehicle dynamometer and also on the road. The corresponding results show that the control algorithm is able to fully exploit the multi-input characteristic of the two-stage turbocharging. The high-pressure stage is used for quick pressure increases, whereas the low-pressure stage is used for high mass flows.

Chapter 10

Conclusions and Outlook

In this thesis, we have proposed a variety of tailored algorithms for embedded optimization, numerical simulation methods and corresponding sensitivity propagation techniques in order to allow for real-time feasible solutions of optimal control problems on embedded hardware.

Embedded optimization and simulation methods For this purpose, we first introduced the general algorithmic techniques that are crucial for the implementation of fast nonlinear model predictive control and estimation schemes in Chapter 1. An important component in any direct optimal control method consists of the integrator, which performs the numerical simulation and corresponding sensitivity analysis for the system of differential equations which describes the process of interest. An overview on explicit and implicit integration schemes and sensitivity propagation techniques has been provided in Chapter 2. This chapter discussed the efficient implementation of implicit Runge-Kutta methods in more detail, e.g., when the Newton-type scheme reuses the Jacobian factorization from the sensitivity computation. The use of a collocation scheme with the continuous output feature has also been motivated based on various optimal control applications. Numerical results based on the `ACADO` code generated integrators show considerable speedups over general-purpose implementations for the targeted real-time applications of optimal control for small to medium-scale systems.

Symmetric Hessian propagation This work focused on Newton-type optimization, more specifically based on sequential quadratic programming algorithms in

the case of inequality constrained problems. For this purpose, first and higher order derivatives are typically needed which can form a major computational effort especially in the case of the numerical simulation of nonlinear system dynamics. Chapter 3 therefore presented a novel symmetric Hessian propagation technique, which allows one to maintain and exploit the symmetry of the Hessian contributions within the propagation scheme. The algorithm has been presented for the context of both a discrete and continuous-time sensitivity analysis. These symmetric equations are additionally shown to allow for a memory efficient three-sweep Hessian propagation, unlike the classical forward-backward scheme. In addition to the reduced memory requirements, numerical experiments showed a computational speedup factor of about 2.

Outlook: The applicability of the proposed symmetric Hessian propagation scheme for the solution of large-scale optimal control problems, and its corresponding advantages for such applications, could be further investigated.

Structure exploitation for dynamic systems When considering embedded applications of optimal control for fast dynamic systems, the use of structure exploiting algorithms can become crucial in order to satisfy the real-time requirements. As motivated in Chapter 4, systems of differential equations and the corresponding integration schemes often allow for the exploitation of linear subsystems. Based on the proposed three-stage model structure, computational speedup factors of about 10 were shown possible for real-world applications. Instead of such a sequential definition of differential equations, one could also have a more general set of interconnected subsystems. Distributed multiple shooting, for example, allows one to treat such decomposable dynamic systems in direct optimal control, based on a suitable parameterization of the coupling variables. Chapter 5 presented a compression algorithm to efficiently exploit the resulting coupling structure in the convex subproblem. Based on this technique, the DMS approach has been illustrated to result in impressive speedups even for a serial implementation.

Outlook: Tailored simulation methods can be developed for a general class of decomposable dynamic systems, represented by a directed graph of (partially) linear and nonlinear subsystems. Based on a modeling environment with differentiation capabilities, one could implement an automatic detection of exploitable structures as a powerful tool for the problem formulation. In doing so, it is however important to preserve a certain level of transparency to the user, on how particular modeling choices affect the computational cost.

Lifted collocation integrators The implementation of an implicit integration method relies on an iterative scheme to solve the nonlinear equations as part of

each integration step. When embedding such an integrator within a Newton-type optimization algorithm, one ends up with an outer and inner level of iterations which is typically not the most efficient situation from a computational point of view. Chapter 6 therefore motivated the use of an implicit variant of the lifted Newton method, which avoids such inner iterations and is applicable to any implicit scheme. More specifically, the chapter presented a novel family of lifted collocation integrators and showed their connection to direct collocation. An extension to inexact Newton implementations has been proposed using either an adjoint differentiation technique or an iterative sensitivity propagation. The latter approach results in a novel optimization algorithm, which we referred to as the inexact Newton scheme based on iterated sensitivities (INIS). The open-source implementation of the lifted integrators within the `ACADO` code generation tool showed a typical speedup factor of about 2 over the standard method without lifting. When using one of the proposed inexact lifted collocation schemes, a further speedup of up to factor 10 has been observed.

Outlook: These computational benefits would be greater for optimal control methods, based on a relatively high order for the collocation polynomials. For example, one could implement a lifted variant of the popular pseudospectral method in which a high order polynomial is defined over the complete horizon.

Inexact Newton with iterated sensitivities (INIS) When using an inexact Newton based optimization algorithm, it is well known how the corresponding approximation of first or higher order derivative information affects the asymptotic convergence rate. Chapter 7 however showed that the INIS scheme offers an important advantage over standard inexact Newton methods, regarding its local convergence properties based on the same Jacobian approximation. Local convergence for the inner scheme is neither sufficient nor necessary for asymptotic contraction of the standard adjoint based inexact Newton implementation. In contrast, the INIS optimization method is shown to have the exact same local asymptotic contraction rate as that of the Newton-type scheme for the forward problem. More specifically, local convergence for the Newton-type method on the forward problem is shown to be necessary, and under mild conditions even sufficient for the asymptotic contraction of the corresponding INIS-type optimization algorithm. In addition, an adjoint-free variant of the INIS scheme has been proposed and it is shown to work very well in practice even though the above connection was only proved for quadratic programming problems in that case.

Outlook: We believe that the INIS scheme allows interesting applications in the general domain of dynamic optimization, e.g., including problems with partial differential equations. The main motivation throughout this thesis has

been the use of inexact Jacobian factorizations for direct collocation. The INIS optimization algorithm could however be used to efficiently implement any direct transcription method.

Open-source ACADO software One practical contribution of this project is that all presented algorithmic developments have been implemented either as part of the open-source `ACADO` code generation package or as a proof of concept using these tools. This was highlighted as part of Chapter 8, which presented the automatic code generation features within the `ACADO Toolkit`.

Outlook: Based on gathered experience, the code generation methodology is however not always necessary to obtain the desired numerical performance in practice. It can be sufficient to perform code generation for those components that form the computational bottleneck, which often correspond to internal linear algebra routines. Alternatively, one could rely on platform specific optimizations for the linear algebra kernels [121, 123] which can be used to build up tailored and efficient algorithms for embedded simulation and optimization methods. Following this approach, a new software project is currently under development which would provide a rapid prototyping environment for embedded optimal control applications based on small, medium or even large-scale dynamic systems. Using clearly defined interfaces for the various algorithmic modules, the maintainability, reusability and extensibility of the code implementations can be secured.

Real-world NMPC applications We additionally discussed some of the real-world control applications, which were made possible by using the open-source `ACADO` code generation tool. Chapter 9 presented a particular optimal control application in more detail, namely that of a two-stage turbocharged gasoline engine. The contribution of this chapter was the development of a real-time feasible nonlinear model predictive control scheme that allows one to meet the tough demands of the two-stage turbocharging structure for gasoline engines. The concept was implemented using an `ACADO` generated solver on the dSpace MicroAutoBox hardware and the resulting system has been validated based on in-vehicle experiments.

Outlook: As the two-stage gasoline turbocharging poses higher demands than diesel or single-stage airpath systems, it can be expected that the presented method would also be applicable for those systems. Regarding the use of these algorithmic techniques on tailored embedded control hardware, there is a potential for heterogeneous computing architectures, which can include different technologies such as a field-programmable gate array (FPGA). The

use of specialized processing units, for the different tasks in online algorithms for dynamic optimization, forms a promising research direction to explore.

Outlook on "Numerical Simulation Methods for Embedded Optimization"
By using a continuation technique for parametric optimization in combination with a shifting strategy to obtain an initial guess for the new optimal control problem from the solution of the previous one, the resulting online algorithm can typically stay within its region of local convergence. This thesis therefore omitted a discussion on globalization strategies, even though the presented algorithmic techniques can be extended to a more general optimization framework including such global convergence guarantees. In addition, direct multiple shooting typically profits from using solvers for differential equations with an efficient step size and order selection. Within a real-time framework for embedded applications, one however often implements multiple shooting using fixed step integrators to result in a deterministic runtime and to satisfy the strict timing requirements. Eventually, it becomes crucial to bridge this current gap between online algorithms, which can be conceptually rather simple but computationally efficient based on tailored techniques, and the state of the art in the field of dynamic optimization and simulation methods. A possible route is the extension of the lifted collocation integrators with variable step size and order selection as well as with globalization strategies.

Bibliography

[1] ADIC. http://www.mcs.anl.gov/research/projects/adic/.

[2] CppAD. http://www.coin-or.org/CppAD.

[3] ACADO Toolkit. http://www.acadotoolkit.org, 2009–2016.

[4] HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk, 2011.

[5] ACADO Toolkit discussion. www.sourceforge.net/p/acado/discussion, 2012–2016.

[6] Albersmeyer, J. *Adjoint-based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, University of Heidelberg, 2010.

[7] Albersmeyer, J., and Bock, H. Sensitivity Generation in an Adaptive BDF-Method. In *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 6-10, 2006, Hanoi, Vietnam* (2008), Springer, pp. 15–24.

[8] Albersmeyer, J., and Diehl, M. The lifted Newton method and its application in optimization. *SIAM Journal on Optimization 20*, 3 (2010), 1655–1684.

[9] Albin, T., Frank, F., Ritter, D., Abel, D., Quirynen, R., and Diehl, M. Nonlinear MPC for combustion engine control: A parameter study for realizing real-time feasibility. In *Proceedings of the IEEE Multi-conference on Systems and Control (MSC)* (2016), pp. 311–316.

[10] Albin, T., Ritter, D., Abel, D., Liberda, N., Quirynen, R., and Diehl, M. Nonlinear MPC for a two-stage turbocharged gasoline engine

airpath. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2015), pp. 849–856.

[11] Albin, T., Ritter, D., Liberda, N., Pischinger, S., and Abel, D. Two-stage turbocharged gasoline engines: Experimental validation of model-based control. *4th IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling (E-COSM) 48*, 15 (2015), 124–131.

[12] Albin, T., Ritter, D., Quirynen, R., and Diehl, M. In-vehicle realization of nonlinear MPC for gasoline two-stage turbocharging airpath control. *IEEE Transactions on Control Systems Technology* (2016). (submitted).

[13] Alessandri, A., Baglietto, M., Battistelli, G., and V.Zavala. Advances in moving horizon estimation for nonlinear systems. 5681–5688.

[14] Alexander, R. Diagonally implicit Runge-Kutta methods for stiff O.D.E.'s. *SIAM Journal on Numerical Analysis 14*, 6 (1977), 1006–1021.

[15] Alexe, M., and Sandu, A. Forward and adjoint sensitivity analysis with continuous explicit Runge-Kutta schemes. *Applied Mathematics and Computation 208*, 2 (2009), 328–346.

[16] Allgower, E. L., and Georg, K. *Introduction to Numerical Continuation Methods*. Colorado State University Press, 1990.

[17] Allgöwer, F., Badgwell, T., Qin, J., Rawlings, J., and Wright, S. Nonlinear Predictive Control and Moving Horizon Estimation – An Introductory Overview. In *Advances in Control, Highlights of ECC'99*. Springer, 1999, pp. 391–449.

[18] Amrit, R., Rawlings, J., and Angeli, D. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control 35*, 2 (2011), 178–186.

[19] Andersson, J. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, K.U. Leuven, October 2013.

[20] Andersson, J., Akesson, J., and Diehl, M. CasADi – a symbolic package for automatic differentiation and optimal control. In *Recent Advances in Algorithmic Differentiation* (2012), vol. 87 of *Lecture Notes in Computational Science and Engineering*, Springer, pp. 297–307.

[21] Arnold, M., Burgermeister, B., and Eichberger, A. Linearly implicit time integration methods in real-time applications: DAEs and stiff ODEs. *Multibody System Dynamics 17*, 2 (2007), 99–117.

[22] Ascher, U., and Petzold, L. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations.* SIAM, Philadelphia, 1998.

[23] Asprion, J. *Optimal Control of Diesel Engines.* PhD thesis, ETH Zurich, 2013.

[24] Axehill, D. Controlling the level of sparsity in MPC. *Systems & Control Letters 76* (2015), 1–7.

[25] Axehill, D., and Morari, M. An alternative use of the Riccati recursion for efficient optimization. *Systems & Control Letters 61*, 1 (2012), 37–40.

[26] Bartels, S. *Numerical Approximation of Partial Differential Equations.* Springer International Publishing, 2016.

[27] Bauer, I., Bock, H., and Schlöder, J. DAESOL – a BDF-code for the numerical solution of differential algebraic equations. Internal report, IWR, SFB 359, University of Heidelberg, 1999.

[28] Baumgarte, J. Stabilization of Constraints and Integrals of Motion in Dynamical Systems. *Computer Methods in Applied Mechanics and Engineering 1*, 1 (1972), 1–16.

[29] Bellman, R. *Dynamic programming.* Princeton University Press, 1957.

[30] Bemporad, A., Borrelli, F., and Morari, M. Model Predictive Control Based on Linear Programming - The Explicit Solution. *IEEE Transactions on Automatic Control 47*, 12 (2002), 1974–1985.

[31] Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. *Robust optimization.* Princeton University Press, 2009.

[32] Bertsekas, D. *Dynamic Programming and Optimal Control*, 3rd ed., vol. 2. Athena Scientific, 2007.

[33] Bertsekas, D., and Shreve, S. *Stochastic Optimal Control: The Discrete Time Case.* Athena Scientific, Belmont, MA, 1996.

[34] Betts, J. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. SIAM, 2010.

[35] Bickart, T. A. An efficient solution process for implicit Runge-Kutta methods. *SIAM Journal on Numerical Analysis 14*, 6 (1977), 1022–1027.

[36] Biegler, L. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering 8*, 3–4 (1984), 243–248.

[37] Biegler, L., Ghattas, O., Heinkenschloss, M., Keyes, D., and van Bloemen Waanders (eds.), B. *Real-Time PDE-Constrained Optimization*. Computational Science and Engineering. SIAM, 2007.

[38] Biegler, L., and Rawlings, J. Optimization approaches to nonlinear model predictive control. In *Proc. 4th International Conference on Chemical Process Control - CPC IV*. AIChE, 1991, pp. 543–571.

[39] Biegler, L. T. *Nonlinear Programming*. MOS-SIAM Series on Optimization. SIAM, 2010.

[40] Bischof, C. H., and Bücker, H. M. Computing derivatives of computer programs. In *Modern Methods and Algorithms of Quantum Chemistry: Proceedings, Second Edition*, vol. 3 of *NIC Series*. NIC-Directors, Jülich, 2000, pp. 315–327.

[41] Blumenschein, J., Schwarzgruber, T., Schmied, R., Passenbrunner, T. E., Waschl, H., and del Re, L. Approximate optimal control of discrete I/O systems with C/GMRES. In *Proceedings of the European Control Conference (ECC)* (July 2015), pp. 104–110.

[42] Bock, H. Numerical Solution of Nonlinear Multipoint Boundary Value Problems with Applications to Optimal Control. *Zeitschrift für Angewandte Mathematik und Mechanik 58*, 7 (1978), 407–454.

[43] Bock, H. Numerical treatment of inverse problems in chemical reaction kinetics. In *Modelling of Chemical Reaction Systems*, K. Ebert, P. Deuflhard, and W. Jäger, Eds., vol. 18 of *Springer Series in Chemical Physics*. Springer, Heidelberg, 1981, pp. 102–125.

[44] Bock, H. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, vol. 183 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1987.

[45] Bock, H., Egartner, W., Kappis, W., and Schulz, V. Practical Shape Optimization for Turbine and Compressor Blades by the Use of PRSQP Methods. *Optimization and Engineering 3*, 4 (2002), 395–414.

[46] Bock, H., Eich, E., and Schlöder, J. Numerical Solution of Constrained Least Squares Boundary Value Problems in Differential-Algebraic Equations. In *Numerical Treatment of Differential Equations*, K. Strehmel, Ed. Teubner, Leipzig, 1988.

[47] Bock, H., and Schlöder, J. Numerical solution of retarded differential equations with state-dependent time lags. *Zeitschrift für Angewandte Mathematik und Mechanik 61* (1981), 269–271.

[48] Bock, H. G. Recent advances in parameter identification techniques for ODE. In *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, 1983, pp. 95–121.

[49] Bock, H. G., Diehl, M., Kostina, E. A., and Schlöder, J. P. Constrained optimal feedback control of systems governed by large differential algebraic equations. In *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2007, pp. 3–22.

[50] Bock, H. G., DIehl, M., Leineweber, D. B., and Schlöder, J. P. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In *Nonlinear Predictive Control* (Basel Boston Berlin, 2000), F. Allgöwer and A. Zheng, Eds., vol. 26 of *Progress in Systems Theory*, Birkhäuser, pp. 246–267.

[51] Bock, H. G., and Plitt, K. J. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the IFAC World Congress* (1984), Pergamon Press, pp. 242–247.

[52] Boggs, P. T., and Tolle, J. W. Sequential quadratic programming. *Acta Numerica* (1995), 1–51.

[53] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3*, 1 (2011), 1–122.

[54] Boyd, S., and Vandenberghe, L. *Convex Optimization*. University Press, Cambridge, 2004.

[55] Brandt-Pollmann, U. *Numerical solution of optimal control problems with implicitly defined discontinuities with applications in engineering*. PhD thesis, IWR, University of Heidelberg, 2004.

[56] Brenan, K. E., Campbell, S. L., and Petzold, L. R. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1987.

[57] Broyden, C. G. Quasi-Newton methods and their application to function minimization. *Maths. Comp. 21* (1967), 368–381.

[58] Bruyninckx, H. Open robot control software: the OROCOS project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (2001), vol. 3, IEEE, pp. 2523–2528.

[59] Bryson, A., and Ho, Y.-C. *Applied Optimal Control*. Wiley, New York, 1975.

[60] Buchner, F., Wedowski, S., Sehr, A., Glueck, S., and Schernus, C. In-vehicle optimization of 2-stage turbocharging for gasoline engines. *International Journal of Automotive Engineering 2*, 4 (2011), 143–148.

[61] Büskens, C., and Maurer, H. SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. *Journal of Computational and Applied Mathematics 120*, 1–2 (2000), 85–108.

[62] Butcher, J. On the implementation of implicit Runge-Kutta methods. *BIT Numerical Mathematics 16*, 3 (1976), 237–240.

[63] Butcher, J. C. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2003.

[64] Cameron, I. T. Solution of differential-algebraic systems using diagonally implicit Runge-Kutta methods. *IMA Journal of Numerical Analysis 3* (1983), 273–289.

[65] Cao, Y., Li, S., and Petzold, L. Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software. *Journal of Computational and Applied Mathematics 149* (2002), 171–191.

[66] Cao, Y., Li, S., Petzold, L., and Serban, R. Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint DAE System and its Numerical Solution. *SIAM Journal on Scientific Computing 24*, 3 (2003), 1076–1089.

[67] Caracotsios, M., and Stewart, W. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Computers and Chemical Engineering. 9* (1985), 359–365.

[68] Chen, H., and Allgöwer, F. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica 34*, 10 (1998), 1205–1218.

[69] Coddington, E., and Levinson, N. *Theory of Ordinary Differential Equations*. McGraw-Hill, New York, 1955.

[70] Cong, N. H., and Xuan, L. N. Parallel-Iterated RK-type PC Methods with Continuous Output Formulas. *International Journal of Computer Mathematics 80*, 8 (2003), 1025–1035.

[71] Cooper, G., and Vignesvaran, R. Some schemes for the implementation of implicit Runge-Kutta methods. *Journal of Computational and Applied Mathematics 45*, 1–2 (1993), 213–225.

[72] Crouch, P., and Grossman, R. Numerical Integration of Ordinary Differential Equations on Manifolds. *Journal of Nonlinear Science 3*, 1 (1993), 1–33.

[73] Curtis, F. E., Johnson, T. C., Robinson, D. P., and Wächter, A. An inexact sequential quadratic optimization algorithm for nonlinear optimization. *SIAM Journal on Optimization 24*, 3 (2014), 1041–1074.

[74] Curtis, F. E., Nocedal, J., and Wächter, A. A matrix-free algorithm for equality constrained optimization problems with rank-deficient Jacobians. *SIAM Journal on Optimization 20*, 3 (Sept. 2009), 1224–1249.

[75] Dahlquist, G. Convergence and stability in numerical integration of ordinary differential equations. *Math. Scand. 4* (1956), 33–53.

[76] Debrouwere, F., Vukov, M., Quirynen, R., Diehl, M., and Swevers, J. Experimental validation of combined nonlinear optimal control and estimation of an overhead crane. In *Proceedings of the IFAC World Congress* (2014), pp. 9617–9622.

[77] Dembo, R., Eisenstat, S., and Steihaug, T. Inexact Newton methods. *SIAM Journal of Numerical Analysis 19*, 2 (April 1982), 400–408.

[78] Demmel, J., Higham, N., and Schreiber, R. Block LU Factorization. *IMA Preprint Series*, 929 (1992).

[79] Dennis, J. E. On Newton-like methods. *Numerische Mathematik 11*, 4 (1968), 324–330.

[80] Dennis, J. E., and Moré, J. J. Quasi-Newton Methods, Motivation and Theory. *SIAM Review 19*, 1 (January 1977), 46–89.

[81] Deuflhard, P. Order and stepsize control in extrapolation methods. *Numerische Mathematik 41* (1983), 399–422.

[82] Deuflhard, P. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, vol. 35. Springer, 2011.

[83] Deuflhard, P., Hairer, E., and Zugck, J. One step and extrapolation methods for differential-algebraic systems. *Numerische Mathematik 51* (1987), 501–516.

[84] Dickinson, P., Glover, K., Collings, N., Yamashita, Y., Yashiro, Y., and Hoshi, T. Real-time control of a two-stage serial VGT diesel engine using MPC. *4th IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling (E-COSM) 48*, 15 (2015), 117–123.

[85] Diehl, M. Airborne wind energy: Basic concepts and physical foundations. In *Airborne Wind Energy*. Springer Berlin Heidelberg, 2013, pp. 3–22.

[86] Diehl, M. *Lecture Notes on Numerical Optimization*. 2016. (Available online: http://cdn.syscop.de/publications/Diehl2016.pdf).

[87] Diehl, M., Amrit, R., and Rawlings, J. B. A Lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control 56*, 3 (March 2011), 703–707.

[88] Diehl, M., Bock, H. G., Schlöder, J., Findeisen, R., Nagy, Z., and Allgöwer, F. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control 12*, 4 (2002), 577–585.

[89] Diehl, M., Bock, H. G., and Schlöder, J. P. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization 43*, 5 (2005), 1714–1736.

[90] Diehl, M., Ferreau, H. J., and Haverbeke, N. Efficient numerical methods for nonlinear MPC and moving horizon estimation. In *Nonlinear model predictive control*, L. Magni, M. Raimondo, and F. Allgöwer, Eds., vol. 384 of *Lecture Notes in Control and Information Sciences*. Springer, 2009, pp. 391–417.

[91] Diehl, M., Findeisen, R., Allgöwer, F., Bock, H. G., and Schlöder, J. P. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl. 152*, 3 (2005), 296–308.

[92] Diehl, M., Leineweber, D., and Schäfer, A. MUSCOD-II Users' Manual. IWR-Preprint 2001-25, University of Heidelberg, 2001.

[93] Diehl, M., Magni, L., and Nicolao, G. D. Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control 28*, 1 (2004), 37–45.

[94] Diehl, M., Walther, A., Bock, H. G., and Kostina, E. An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization. *Optimization Methods and Software 25*, 4 (2010), 531–552.

[95] Domahidi, A., Chu, E., and Boyd, S. ECOS: An SOCP solver for embedded systems. In *Proceedings of the European Control Conference (ECC)* (2013), IEEE, pp. 3071–3076.

[96] Domahidi, A., and Perez, J. FORCES professional. embotech GmbH (`http://embotech.com/FORCES-Pro`), July 2014.

[97] Domahidi, A., Zgraggen, A., Zeilinger, M., Morari, M., and Jones, C. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (Maui, HI, USA, Dec. 2012), pp. 668–674.

[98] Driver, R. D. *Ordinary and Delay Differential Equations*, vol. 20. Springer New York, 1977.

[99] Eich, E. Numerische Behandlung semi-expliziter differentiell-algebraischer Gleichungssysteme vom Index I mit BDF Verfahren. Master's thesis, Universität Bonn, Bonn, 1987.

[100] Emekli, M. E., and Güvenç, B. A. Explicit MIMO model predictive boost pressure control of a two-stage turbocharged diesel engine. *IEEE Transactions on Control Systems Technology* (2016).

[101] Enright, W. H., Higham, D. J., Owren, B., and Sharp, P. W. A survey of the explicit Runge-Kutta method. Tech. rep., 1995.

[102] Enright, W. H., Jackson, K. R., Nørsett, S. P., and Thomsen, P. G. Interpolants for Runge-Kutta formulas. *ACM Trans. Math. Softw. 12* (1986), 193–218.

[103] Eriksson, L., and Nielsen, L. Modeling and control of engines and drivelines. *Wiley* (2014).

[104] Fabien, B. dsoa: The implementation of a dynamic system optimization algorithm. *Optimal Control Applications and Methods 31* (2010), 231–247.

[105] Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., and Hrovat, D. Integrated braking and steering model predictive control approach in autonomous vehicles. In *Advances in Automotive Control* (2007), vol. 5, pp. 273–278.

[106] Feehery, W. F., Tolsma, J. E., and Barton, P. I. Efficient sensitivity analysis of large-scale differential-algebraic systems. *Applied Numerical Mathematics 25* (1997), 41–54.

[107] Ferreau, H., Kraus, T., Vukov, M., Saeys, W., and Diehl, M. High-speed moving horizon estimation based on automatic code generation. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2012), pp. 687–692.

[108] Ferreau, H. J. *Model Predictive Control Algorithms for Applications with Millisecond Timescales*. PhD thesis, K.U. Leuven, 2011.

[109] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation 6*, 4 (2014), 327–363.

[110] Ferreau, H. J., Ortner, P., Langthaler, P., del Re, L., and Diehl, M. Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control 31*, 2 (2007), 293–301.

[111] Findeisen, R., and Allgöwer, F. Computational Delay in Nonlinear Model Predictive Control. Proc. Int. Symp. Adv. Control of Chemical Processes, ADCHEM, 2003.

[112] Fletcher, R. *Practical Methods of Optimization*, 2nd ed. Wiley, Chichester, 1987.

[113] Floudas, C., Akrotirianakis, I., Caratzoulas, S., Meyer, C., and Kallrath, J. Global optimization in the 21st century: Advances and challenges. *Computers and Chemical Engineering 29*, 6 (2005), 1185–1202.

[114] Franklin, G., Powell, J., and Emami-Naeini, A. *Feedback control of dynamic systems*, 6 ed. Prentice Hall, 2009.

[115] Frasch, J. *Parallel Algorithms for Optimization of Dynamic Systems in Real-Time*. PhD thesis, KU Leuven and U Magdeburg, 2014.

[116] Frasch, J. V., Gray, A. J., Zanon, M., Ferreau, H. J., Sager, S., Borrelli, F., and Diehl, M. An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In *Proceedings of the European Control Conference (ECC)* (2013), pp. 4136–4141.

[117] Frasch, J. V., Sager, S., and Diehl, M. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computations 7*, 3 (2015), 289–329.

[118] Frasch, J. V., Wirsching, L., Sager, S., and Bock, H. G. Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control* (2012), vol. 45, pp. 138–144.

[119] Frehse, J. Existence of optimal controls I. *Operations Research Verfahren 31* (1979), 213–225.

[120] Freund, R., and Jarre, F. A sensitivity analysis and a convergence result for a sequential semidefinite programming method. Tech. rep., Bell Laboratories, Murray Hill, 2003.

[121] Frison, G. *Algorithms and Methods for High-Performance Model Predictive Control*. PhD thesis, Technical University of Denmark (DTU), 2015.

[122] Frison, G., and Jørgensen, J. B. A fast condensing method for solution of linear-quadratic control problems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2013), pp. 7715–7720.

[123] Frison, G., Sorensen, H. B., Dammann, B., and Jørgensen, J. B. High-performance small-scale solvers for linear model predictive control. In *Proceedings of the European Control Conference (ECC)* (June 2014), pp. 128–133.

[124] Fruzzetti, K., Palazoglu, A., and McDonald, K. Nonlinear model predictive control using Hammerstein models. *Journal of Process Control 7*, 1 (1997), 31–41.

[125] Gao, Y., Gray, A., Frasch, J. V., Lin, T., Tseng, H. E., Hedrick, J., and Borrelli, F. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control* (2012).

[126] Garg, D., Patterson, M. A., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica 46*, 11 (2010), 1843–1851.

[127] Geebelen, K. *Design and Operation of Airborne Wind Energy Systems*. PhD thesis, K.U. Leuven, 2015.

[128] Geebelen, K., Ahmad, H., Vukov, M., Gros, S., Swevers, J., and Diehl, M. An experimental test set-up for launch/recovery of an airborne wind energy (AWE) system. In *Proceedings of the American Control Conference (ACC)* (2012), pp. 4405–4410.

[129] Geebelen, K., Vukov, M., Wagner, A., Ahmad, H., Zanon, M., Gros, S., Vandepitte, D., Swevers, J., and Diehl, M. An experimental test setup for advanced estimation and control of an airborne wind energy systems. In *Airborne Wind Energy*, U. Ahrens, M. Diehl, and R. Schmehl, Eds. Springer, 2013, pp. 459–471.

[130] Geebelen, K., Wagner, A., Gros, S., Swevers, J., and Diehl, M. Moving horizon estimation with a Huber penalty function for robust pose estimation of tethered airplanes. In *Proceedings of the American Control Conference (ACC)* (2013), pp. 6169–6174.

[131] Gerdts, M. *Optimal Control of ODEs and DAEs*. Berlin, Boston: De Gruyter, 2011.

[132] Gertz, E. M., and Wright, S. J. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software 29*, 1 (2003), 58–81.

[133] Gill, P., Murray, W., and Saunders, M. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review 47*, 1 (2005), 99–131.

[134] Gill, P. E., and Robinson, D. P. A primal-dual augmented Lagrangian. *Computational Optimization and Applications 51*, 1 (2012), 1–25.

[135] Giselsson, P. Improved fast dual gradient methods for embedded model predictive control. In *Proceedings of the 2014 IFAC World Congress* (2014), vol. 47, pp. 2303–2309.

[136] Glueck, S. Charging concepts for a two-stage turbocharging gasoline engine. *PhD Thesis RWTH Aachen University* (2013).

[137] Golub, G., and Loan, C. *Matrix Computations*, 3rd ed. Johns Hopkins University Press, Baltimore, 1996.

[138] González-Pinto, S., Montijano, J. I., and Rández, L. Iterative schemes for three-stage implicit Runge-Kutta methods. *Appl. Numer. Math. 17*, 4 (Aug. 1995), 363–382.

[139] González-Pinto, S., Pérez-Rodríguez, S., and Montijano, J. I. Implementation of high-order implicit Runge-Kutta methods. *Computers & Mathematics with Applications 41*, 7-8 (2001), 1009–1024.

[140] Gower, R. M., and Mello, M. P. A new framework for the computation of hessians. *Optimization Methods and Software 27*, 2 (2012), 251–273.

[141] Griewank, A. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. No. 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.

[142] Griewank, A., Juedes, D., and Utke, J. Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Softw. 22*, 2 (June 1996), 131–167.

[143] Griewank, A., and Walther, A. On Constrained Optimization by Adjoint based quasi-Newton Methods. *Optimization Methods and Software 17* (2002), 869–889.

[144] Gros, S., and Diehl, M. NMPC based on Huber penalty functions to handle large deviations of quadrature states. In *Proceedings of the American Control Conference (ACC)* (2013), pp. 3159–3164.

[145] Gros, S., Zanon, M., and Diehl, M. Control of airborne wind energy systems based on nonlinear model predictive control & moving horizon estimation. In *Proceedings of the European Control Conference (ECC)* (2013), pp. 1017–1022.

[146] Gros, S., Zanon, M., and Diehl, M. Baumgarte stabilisation over the SO(3) rotation group for control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2015), pp. 620–625.

[147] Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control* (2016).

[148] Gros, S., Zanon, M., Vukov, M., and Diehl, M. Nonlinear MPC and MHE for Mechanical Multi-Body Systems with Application to Fast Tethered Airplanes. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, Noordwijkerhout, The Netherlands* (2012), pp. 86–93.

[149] Grüne, L. NMPC Without Terminal Constraints. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control 2012* (2012).

[150] Grüne, L. Economic receding horizon control without terminal constraints. *Automatica 49* (2013), 725–734.

[151] Grüne, L., and Pannek, J. *Nonlinear Model Predictive Control*. Springer, London, 2011.

[152] Guddat, J., Vasquez, F. G., and Jongen, H. *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Teubner, Stuttgart, 1990.

[153] Guzzella, L., and Onder, C. Introduction to modelling and control of internal combustion engine systems. *Springer* (2004).

[154] Hager, W. W. Rates of convergence for discrete approximations to unconstrained control problems. *SIAM Journal on Numerical Analysis 13*, 4 (1976), 449–472.

[155] Hager, W. W. Runge-Kutta methods in optimal control and the transformed adjoint system. *Numerische Mathematik 87*, 2 (2000), 247–282.

[156] Hairer, E., Lubich, C., and Wanner, G. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer, 2006.

[157] Hairer, E., Nørsett, S., and Wanner, G. *Solving Ordinary Differential Equations I*, 2nd ed. Springer Series in Computational Mathematics. Springer, Berlin, 1993.

[158] Hairer, E., and Wanner, G. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, 2nd ed. Springer, Berlin Heidelberg, 1991.

[159] Hannemann-Tamás, R., and Imsland, L. S. Full algorithmic differentiation of a rosenbrock-type method for direct single shooting. In *European Control Conference, ECC 2014, Strasbourg, France, June 24-27, 2014* (2014), pp. 1242–1248.

[160] Hargraves, C., and Paris, S. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance 10*, 4 (1987), 338–342.

[161] Haseltine, E., and Rawlings, J. Critical Evaluation of Extended Kalman Filtering and Moving-Horizon Estimation. *Industrial and Engineering Chemistry Research 44* (2005), 2451–2460.

[162] Heinkenschloss, M., and Vicente, L. Analysis of Inexact Trust-Region SQP Algorithms. *SIAM Journal on Optimization 12*, 2 (2001), 283–302.

[163] Herceg, M., Kvasnica, M., Jones, C., and Morari, M. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*

(Zürich, Switzerland, July 17–19 2013), pp. 502–510. http://control.ee.ethz.ch/~mpt.

[164] Herceg, M., Raff, T., Findeisen, R., and Allgower, F. Nonlinear model predictive control of a turbocharged diesel engine. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control* (2006), IEEE, pp. 2766–2771.

[165] Hettich, R., and Jongen, H. *Semi-infinite programming: Conditions of optimality and applications*, vol. 7. Springer Berlin Heidelberg, 1978.

[166] Hindmarsh, A., Brown, P., Grant, K., Lee, S., Serban, R., Shumaker, D., and Woodward, C. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software 31*, 3 (2005), 363–396.

[167] Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S. *Optimization with PDE constraints*. Springer, 2009.

[168] Hochbruck, M., and Ostermann, A. Exponential integrators. *Acta Numerica 19* (2010), 209–286.

[169] Hochbruck, M., Ostermann, A., and Schweitzer, J. Exponential rosenbrock-type methods. *SIAM Journal of Numerical Analysis 47*, 1 (2009), 786–803.

[170] Hoffmann, G., Waslander, H. H. S., and Tomlin, C. Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment. In *AIAA Guidance, Navigation and Control Conference and Exhibit* (2007).

[171] Horn, M. K. Fourth- and fifth-order, scaled Runge-Kutta algorithms for treating dense output. *SIAM Journal on Numerical Analysis 20*, 3 (1983), 558–568.

[172] Hours, J.-H., and Jones, C. N. A parametric nonconvex decomposition algorithm for real-time and distributed NMPC. *IEEE Transactions on Automatic Control 61*, 2 (2014), 287–302.

[173] Houska, B. *Robust Optimization of Dynamic Systems*. PhD thesis, Katholieke Universiteit Leuven, 2011. (ISBN: 978-94-6018-394-2).

[174] Houska, B., and Chachuat, B. Branch-and-lift algorithm for deterministic global optimization in nonlinear optimal control. *Journal of Optimization Theory and Applications 162*, 1 (2014), 208–248.

[175] Houska, B., and Diehl, M. A quadratically convergent inexact SQP method for optimal control of differential algebraic equations. *Optimal Control Applications and Methods 34*, 4 (2013), 396–414.

[176] Houska, B., Ferreau, H. J., and Diehl, M. ACADO toolkit – an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods 32*, 3 (2011), 298–312.

[177] Houska, B., Ferreau, H. J., and Diehl, M. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica 47*, 10 (2011), 2279–2285.

[178] Houska, B., Frasch, J. V., and Diehl, M. An augmented Lagrangian based algorithm for distributed non-convex optimization. *SIAM Journal on Optimization 26*, 2 (2016), 1101–1127.

[179] Huang, M., Nakada, H., Butts, K., and Kolmanovsky, I. Nonlinear model predictive control of a diesel engine air path: A comparison of constraint handling and computational strategies. *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC) 48*, 23 (2015), 372–379.

[180] Huber, P. J. *Robust Statistics*. Wiley, 1981.

[181] Jaeger, H., and Sachs, E. Global convergence of inexact reduced SQP methods. *Optimization Methods and Software 7*, 2 (1997), 83–110.

[182] Jerez, J. L., Goulart, P. J., Richter, S., Constantinides, G. A., Kerrigan, E. C., and Morari, M. Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control 59*, 12 (Dec 2014), 3238–3251.

[183] Jones, C., and Morari, M. Polytopic approximation of explicit model predictive controllers. *IEEE Transactions on Automatic Control 55*, 11 (2010), 2542–2553.

[184] Joshi, A. *Elements of Group Theory for Physicists*. New Age International Publishers, 1997.

[185] Kalmari, J., Backman, J., and Visala, A. A toolkit for nonlinear model predictive control using gradient projection and code generation. *Control Engineering Practice 39* (2015), 56–66.

[186] Kang, J., Cao, Y., Word, D. P., and Laird, C. D. An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition. *Computers & Chemical Engineering 71* (2014), 563–573.

[187] Khalil, H. *Nonlinear Systems*, second ed. Prentice Hall, Upper Saddle River, NJ, 1996.

[188] Kirches, C. A Numerical Method for Nonlinear Robust Optimal Control with Implicit Discontinuities and an Application to Powertrain Oscillations. Diploma thesis, University of Heidelberg, October 2006.

[189] Kirches, C., Bock, H., Schlöder, J., and Sager, S. Block structured quadratic programming for the direct multiple shooting method for optimal control. *Optimization Methods and Software 26* (April 2010), 239–257.

[190] Kirches, C., Wirsching, L., Sager, S., and Bock, H. Efficient numerics for nonlinear model predictive control. In *Recent Advances in Optimization and its Applications in Engineering*. Springer, 2010, pp. 339–357.

[191] Kotman, P., Bitzer, M., and Kugi, A. Flatness-based feedforward control of a two-stage turbocharged diesel air system with EGR. In *2010 IEEE international conference on control applications* (2010), IEEE, pp. 979–984.

[192] Kouzoupis, D., Quirynen, R., Frasch, J. V., and Diehl, M. Block condensing for fast nonlinear MPC with the dual Newton strategy. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)* (2015), vol. 48, pp. 26–31.

[193] Kouzoupis, D., Quirynen, R., Girrbach, F., and Diehl, M. An efficient SQP algorithm for moving horizon estimation with Huber penalties and multi-rate measurements. In *Proceedings of the IEEE Multi-conference on Systems and Control (MSC)* (2016), pp. 1482–1487.

[194] Kouzoupis, D., Quirynen, R., Houska, B., and Diehl, M. A block based ALADIN scheme for highly parallelizable direct optimal control. In *Proceedings of the American Control Conference (ACC)* (2016), pp. 1124–1129.

[195] Kozma, A., Andersson, J., Savorgnan, C., and Diehl, M. Distributed multiple shooting for optimal control of large interconnected systems. In *Proceedings of the International Symposium on Advanced Control of Chemical Processes* (2012), vol. 45, pp. 143–147.

[196] Kozma, A., Conte, C., and Diehl, M. Benchmarking large-scale distributed convex quadratic programming algorithms. *Optimization Methods & Software 30*, 1 (2015), 191–214.

[197] Kraus, T. Real-Time State And Parameter Estimation for NMPC-Based Feedback Control With Application To The Tennessee Eastman Benchmark Process. Master's thesis, University of Heidelberg, 2007.

[198] Kraus, T., Ferreau, H. J., Kayacan, E., Ramon, H., Baerdemaeker, J. D., Diehl, M., and Saeys, W. Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. *Computers and Electronics in Agriculture 98* (October 2013), 25–33.

[199] Kristensen, M., Jørgensen, J., Thomsen, P., and Jørgensen, S. An ESDIRK method with sensitivity analysis capabilities. *Computers and Chemical Engineering 28* (2004), 2695–2707.

[200] Kühl, P., Diehl, M., Kraus, T., Schlöder, J. P., and Bock, H. G. A real-time algorithm for moving horizon state and parameter estimation. *Computers and Chemical Engineering 35*, 1 (2011), 71–83.

[201] Kumar, G. P., Sastry, I. V. K. S., and Cidambaram, M. Periodic operation of a bioreactor with input multiplicities. *The Canadian Journal of Chemical Engineering 71* (1993), 766–770.

[202] Kvasnica, M., Rauova, I., and Miroslav, F. Automatic code generation for real-time implementation of model predictive control. In *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design, Yokohama, Japan* (2010), pp. 993–998.

[203] Lambert, J. D. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[204] Lebiedz, D., Skanda, D., and Fein, M. *Automatic Complexity Analysis and Model Reduction of Nonlinear Biochemical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 123–140.

[205] Lee, T., McClamroch, N. H., and Leok, M. A Lie group variational integrator for the attitude dynamics of a rigid body with application to the 3D pendulum. In *Proceedings of the 2005 IEEE Conference on Control Applications* (2005), pp. 962–967.

[206] Leibfritz, F., and Sachs, E. W. Inexact SQP interior point methods and large scale optimal control problems. *SIAM Journal on Control and Optimization 38*, 1 (2006), 272–293.

[207] Leimkuhler, B., and Reich, S. *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2005.

[208] Leineweber, D. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, vol. 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[209] Leineweber, D., Bauer, I., Schäfer, A., Bock, H., and Schlöder, J. An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II). *Computers and Chemical Engineering 27* (2003), 157–174.

[210] Li, S., Petzold, L., and Zhu, W. Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem. *Applied Numerical Mathematics 32*, 2 (2000), 161–174.

[211] Li, W., and Biegler, L. Multistep, Newton-Type Control Strategies for Constrained Nonlinear Processes. *Chem. Eng. Res. Des. 67* (1989), 562–577.

[212] Li, W., and Biegler, L. Newton-Type Controllers for Constrained Nonlinear Processes with Uncertainty. *Industrial and Engineering Chemistry Research 29* (1990), 1647–1657.

[213] Liniger, A., Domahidi, A., and Morari, M. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods 36*, 5 (2015), 628–647.

[214] Lions, P. *Generalized Solutions of Hamilton-Jacobi Equations*. Pittman, 1982.

[215] Liu, F., Hager, W. W., and Rao, A. V. Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction. *Journal of the Franklin Institute 352*, 10 (2015), 4081–4106.

[216] Ljung, L. *System identification: Theory for the User*. Prentice Hall, Upper Saddle River, N.J., 1999.

[217] Loyd, M. Crosswind Kite Power. *Journal of Energy 4*, 3 (July 1980), 106–111.

[218] Luenberger, D. *Introduction to Dynamic Systems*. Wiley, 1979.

[219] Magni, L., De Nicolao, G., Magnani, L., and Scattolini, R. A stabilizing model-based predictive control for nonlinear systems. *Automatica 37*, 9 (2001), 1351–1362.

[220] Maly, T., and Petzold, L. R. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics 20*, 1–2 (February 1996), 57–79.

[221] Mattingley, J., and Boyd, S. *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2009, ch. Automatic Code Generation for Real-Time Convex Optimization, pp. 1–41.

[222] Mattingley, J., Wang, Y., and Boyd, S. Code generation for receding horizon control. In *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design* (Yokohama, Japan, 2010), pp. 985–992.

[223] Michelsen, M. L. Semi-implicit Runge-Kutta methods for stiff systems: program description and application examples. *Inst. f. Kemiteknik, Danmarks tekniske Højskole, Lynby* (1976).

[224] Michiels, W., and Niculescu, S.-I. *Stability and stabilization of time-delay systems. An eigenvalue based approach*, vol. 12 of *Advances in Design and Control*. SIAM, 2007.

[225] Moraal, P., and Grizzle, J. Observer design for nonlinear systems with discrete-time measurements. *Automatic Control, IEEE Transactions on 40*, 3 (1995), 395–404.

[226] Morrison, D. D., Riley, J. D., and Zancanaro, J. F. Multiple shooting method for two-point boundary value problems. *Communications of the ACM 5*, 12 (1962), 613–614.

[227] Moulin, P., and Chauvin, J. Modelling and control of the air system of a turbocharged gasoline engine. *Journal of Control Engineering Practice Control* (2011), 287–297.

[228] Müller, M. A., Angeli, D., and Allgöwer, F. On necessity and robustness of dissipativity in economic model predictive control. *IEEE Transactions on Automatic Control 60*, 6 (2015), 1671–1676.

[229] Murilo, A., Alamir, M., and Alberer, D. A general NMPC framework for a diesel engine air path. *International Journal of Control 87*, 10 (2014), 2194–2207.

[230] Nesterov, Y. *Introductory lectures on convex optimization: a basic course*, vol. 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

[231] Nicolao, G., Magni, L., and Scattolini, R. Stabilizing Receding-Horizon control of nonlinear time varying systems. *IEEE Transactions on Automatic Control AC-43*, 7 (1998), 1030–1036.

[232] Nocedal, J., and Wright, S. J. *Numerical Optimization*, 2 ed. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[233] Norsett, S. P. Semi-explicit Runge-Kutta methods. Tech. Rep. 6, Department of Mathematics, University of Trondheim, 1974.

[234] O'Donoghue, B., Stathopoulos, G., and Boyd, S. A Splitting Method for Optimal Control. *IEEE Transactions on Control Systems Technology 21*, 6 (2013), 2432–2442.

[235] Ohtsuka, T. A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica 40*, 4 (2004), 563–574.

[236] Ohtsuka, T. A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control. In *Control Conference (ECC), 2015 European* (July 2015), pp. 73–86.

[237] Ohtsuka, T., and Kodama, A. Automatic code generation system for nonlinear receding horizon control. *Transactions of the Society of Instrument and Control Engineers 38*, 7 (2002), 617–623.

[238] Oohori, T., and Ohuchi, A. An efficient implementation of Karmarkar's algorithm for large sparse linear programs. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics,* (1988), vol. 2, pp. 1389–1392.

[239] Osborne, M. On shooting methods for boundary value problems. *Journal of Mathematical Analysis and Applications 27* (1969), 417–433.

[240] Özyurt, D. B., and Barton, P. I. Cheap Second Order Directional Derivatives of Stiff ODE Embedded Functionals. *SIAM Journal on Scientific Computing 26* (2005), 1725–1743.

[241] Pantelides, C., Sargent, R., and Vassiliadis, V. Optimal control of multistage systems described by high-index differential-algebraic equations. In *Computational Optimal Control*, R. Bulirsch and D. Kraft, Eds. Birkhäuser, Basel, 1994, pp. 177–191.

[242] Papastavridis, J. *Analytical Mechanics*. Oxford University Press, Inc., 2002.

[243] Parikh, N., and Boyd, S. Block splitting for distributed optimization. *Mathematical Programming Computation 6* (2012), 77—-102.

[244] Parulekar, S. J. Analysis of forced periodic operations of continuous bioprocesses - single input variations. *Chemical Engineering Science 53(14)* (1998), 2481–2502.

[245] Patrinos, P., and Bemporad, A. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *Automatic Control, IEEE Transactions on 59*, 1 (Jan 2014), 18–33.

[246] Patterson, M. A., Hager, W. W., and Rao, A. V. A ph mesh refinement method for optimal control. *Optimal Control Applications and Methods 36*, 4 (2015), 398–421.

[247] Patterson, M. A., and Rao, A. V. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Trans. Math. Softw. 41*, 1 (Oct. 2014), 1–37.

[248] Pesce, C. P. The Application of Lagrange Equations to Mechanical Systems With Mass Explicitly dependent on Position. *Journal of Applied Mechanics 70* (September 2003), 751–756.

[249] Plitt, K. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Master's thesis, Universität Bonn, 1981.

[250] Polak, E. *Computational methods in optimization: a unified approach.* Academic Press, New York„ 1971.

[251] Pontryagin, L., Boltyanski, V., Gamkrelidze, R., and Miscenko, E. *The Mathematical Theory of Optimal Processes.* Wiley, Chichester, 1962.

[252] Potschka, A. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints.* PhD thesis, University of Heidelberg, 2011.

[253] Potschka, A., Bock, H., Engell, S., Küpper, A., and Schlöder, J. A Newton-Picard inexact SQP method for optimization of SMB processes. Tech. rep., 2008.

[254] Potschka, A., Bock, H. G., and Schlöder, J. P. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software 24*, 2 (2009), 237–252.

[255] Powell, M. J. D. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical Analysis, Dundee 1977* (Berlin, 1978), G. A. Watson, Ed., vol. 630 of *Lecture Notes in Mathematics*, Springer, pp. 144–157.

[256] Prothero, A., and Robinson, A. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation 28*, 125 (1974), 145–162.

[257] Qin, S., and Badgwell, T. An overview of nonlinear model predictive control applications. In *Nonlinear Predictive Control* (Basel Boston Berlin, 2000), F. Allgöwer and A. Zheng, Eds., vol. 26 of *Progress in Systems Theory*, Birkhäuser, pp. 370–392.

[258] Qiu, Z., Santillo, M., Jankovic, M., and Sun, J. Composite adaptive internal model control and its application to boost pressure control of a turbocharged gasoline engine. *IEEE Transactions on Control Systems Technology 23*, 6 (2015), 2306–2315.

[259] Quirynen, R. Automatic code generation of Implicit Runge-Kutta integrators with continuous output for fast embedded optimization. Master's thesis, KU Leuven, 2012.

[260] Quirynen, R., Gros, S., and Diehl, M. Efficient NMPC for nonlinear models with linear subsystems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2013), pp. 5101–5106.

[261] Quirynen, R., Gros, S., and Diehl, M. Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. In *Proceedings of the European Control Conference (ECC)* (2013), pp. 3077–3082.

[262] Quirynen, R., Gros, S., and Diehl, M. Inexact Newton based lifted implicit integrators for fast nonlinear MPC. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)* (2015), pp. 32–38.

[263] Quirynen, R., Gros, S., and Diehl, M. Lifted implicit integrators for direct optimal control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2015), pp. 3212–3217.

[264] Quirynen, R., Gros, S., and Diehl, M. Inexact Newton-type optimization with iterated sensitivities. *SIAM Journal on Optimization (accepted, preprint available at Optimization Online, 2016-06-5502)* (2016).

[265] Quirynen, R., Gros, S., Houska, B., and Diehl, M. Lifted collocation integrators for direct optimal control in ACADO toolkit. *Mathematical Programming Computation (accepted, preprint available at Optimization Online, 2016-05-5468)* (2016).

[266] Quirynen, R., Houska, B., and Diehl, M. Efficient symmetric hessian propagation for direct optimal control. *Journal of Process Control (accepted, preprint available at Optimization Online, 2016-05-5467)* (2016).

[267] Quirynen, R., Houska, B., and Diehl, M. Symmetric hessian propagation for lifted collocation integrators in direct optimal control. In *Proceedings of the American Control Conference (ACC)* (2016), pp. 1117–1123.

[268] Quirynen, R., Houska, B., Vallerio, M., Telen, D., Logist, F., Impe, J. V., and Diehl, M. Symmetric algorithmic differentiation based exact Hessian SQP method and software for economic MPC. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2014), pp. 2752–2757.

[269] Quirynen, R., Vukov, M., and Diehl, M. Auto generation of implicit integrators for embedded NMPC with microsecond sampling times. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference* (2012), M. Lazar and F. Allgöwer, Eds., pp. 175–180.

[270] Quirynen, R., Vukov, M., and Diehl, M. Multiple shooting in a microsecond. In *Multiple Shooting and Time Domain Decomposition Methods*. Springer, 2015, pp. 183–201.

[271] Quirynen, R., Vukov, M., Zanon, M., and Diehl, M. Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators. *Optimal Control Applications and Methods 36* (2014), 685–704.

[272] Quirynen, R., Zanon, M., Kozma, A., and Diehl, M. A compression algorithm for real-time distributed nonlinear MPC. In *Proceedings of the European Control Conference (ECC)* (2015), pp. 3422–3427.

[273] Rao, C. *Moving Horizon Estimation of Constrained and Nonlinear Systems*. PhD thesis, University of Wisconsin–Madison, 2000.

[274] Rao, C., Wright, S., and Rawlings, J. Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications 99* (1998), 723–757.

[275] Rao, C. V., Rawlings, J. B., and Mayne, D. Q. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Transactions on Automatic Control 48*, 2 (2003), 246–258.

[276] Rawlings, J., and Ji, L. Optimization-based State Estimation: Current Status and Some New Results. *Journal of Process Control 22*, 8 (2012), 1439–1444.

[277] Rawlings, J., and Mayne, D. *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[278] Rawlings, J. B., and Amrit, R. Optimizing process economic performance using model predictive control. In *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Springer Berlin Heidelberg, 2009, pp. 119–138.

[279] Reich, S. On an existence and uniqueness theory for nonlinear differential-algebraic equations. *Circuits Syst. Signal Process. 10*, 3 (May 1991), 343–359.

[280] Rheinboldt, W. Differential-algebraic systems as differential equations on manifolds. *Math. of Comput. 43* (1984), 473–482.

[281] Riaza, R. *Differential-Algebraic Systems: Analytical Aspects and Circuit Applications*. World Scientific, 2008.

[282] Richter, S. *Computational complexity certification of gradient methods for real-time model predictive control*. PhD thesis, ETH Zürich, 2012.

[283] Richter, S., Jones, C. N., and Morari, M. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control 57*, 6 (June 2012), 1391–1403.

[284] Robinson, S. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming 7* (1974), 1–16.

[285] Romanenko, A., Pedrosa, N., Leal, J., and Santos, L. *Seminario de Aplicaciones Industriales de Control Avanzado*. 2007, ch. A Linux Based Nonlinear Model Predictive Control Framework, pp. 229–236.

[286] Ruan, L., and Chen, X. Comparison of Several Periodic Operations of a Continous Fermentation Process. *Biotechnol. Prog. 12* (1996), 286–288.

[287] Sandu, A. *Computational Science – ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part IV*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, ch. On the Properties of Runge-Kutta Discrete Adjoints, pp. 550–557.

[288] Sargent, R., and Sullivan, G. The development of an efficient optimal control package. In *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2* (Heidelberg, 1978), J. Stoer, Ed., Springer, pp. 158–168.

[289] Savorgnan, C., Kozma, A., Andersson, J., and Diehl, M. Adjoint-based distributed multiple shooting for large-scale systems. In *18th IFAC World Congress* (2011), vol. 18, pp. 410–415.

[290] Savorgnan, C., Romani, C., Kozma, A., and Diehl, M. Multiple shooting for distributed systems with applications in hydro electricity production. *Journal of Process Control 21* (2011), 738–745.

[291] Schlegel, M., Marquardt, W., Ehrig, R., and Nowak, U. Sensitivity analysis of linearly-implicit differential-algebraic systems by one-step extrapolation. *Applied Numerical Mathematics 48*, 1 (2004), 83–102.

[292] Schlöder, J. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*, vol. 187 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1988.

[293] Schoukens, J., and Pintelon, R. *Identification of Linear Systems: A Practical Guide to Accurate Modeling*. Pergamon Press, 1991.

[294] Schulz, V., Bock, H., and Steinbach, M. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs. *SIAM Journal on Scientific Computing 19* (1998), 440–467.

[295] Schwerin, R., and Winckler, M. Some Aspects of Sensitivity Analysis in Vehicle System Dynamics. *Zeitschrift für Angewandte Mathematik und Mechanik 76*, S3 (1996), 155–158.

[296] Shahzad, A., Kerrigan, E., and Constantinides, G. A warm-start interior-point method for predictive control. Tech. rep., Imperial College London, 2010.

[297] Shuster, M. D. A Survey of Attitude Representations. *The Journal of the Astronautical Science 41*, 4 (1993), 439–517.

[298] Simon, L., Nagy, Z., and Hungerbuehler, K. *Nonlinear Model Predictive Control*, vol. 384 of *Lecture Notes in Control and Information Sciences*. Springer, 2009, ch. Swelling Constrained Control of an Industrial Batch Reactor Using a Dedicated NMPC Environment: OptCon, pp. 531–539.

[299] Stathopoulos, G., Szucs, A., Pu, Y., and Jones, C. Splitting methods in control. In *Proceedings of the European Control Conference (ECC)* (2014), pp. 2478–2483.

[300] Telen, D., Logist, F., Quirynen, R., Houska, B., Diehl, M., and Impe, J. V. Optimal experiment design for nonlinear dynamic (bio)chemical systems using sequential semidefinite programming. *AIChE (American Institute of Chemical Engineers) Journal 60*, 5 (2014), 1728–1739.

[301] Thomasson, A., Leufven, O., Criscuolo, I., and Eriksson, L. Modelling and validation of a boost pressure actuation system, for a series sequentially turbocharged SI engine. *Journal of Control Engineering Practice 21* (2013), 1860–1870.

[302] Thomée, V. *Galerkin Finite Element Methods for Parabolic Problems*, vol. 25. Springer Berlin Heidelberg, 2006.

[303] Tomlab Optimization. PROPT: Matlab Optimal Control Software (ODE,DAE). http://tomdyn.com, 2009–2011.

[304] Tran-Dinh, Q. *Sequential Convex Programming and Decomposition Approaches for Nonlinear Optimization*. Phd thesis, Arenberg Doctoral School, KU Leuven, November 2012.

[305] Tran-Dinh, Q., and Diehl, M. Local convergence of sequential convex programming for nonconvex optimization. In *Recent advances in optimization and its application in engineering*, M.Diehl, F.Glineur, E. Jarlebring, and W. Michiels, Eds. Springer-Verlag, 2010, pp. 93–103.

[306] Ullmann, F. FiOrdOs: A Matlab toolbox for C-code generation for first order methods. Master's thesis, ETH Zurich, December 2011.

[307] Varah, J. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Mathematics of Computation 26*, 120 (1972), 859–868.

[308] Verschueren, R. Design and Implementation of a Time-Optimal Controller for Model Race Cars. Master's thesis, KU Leuven, 2014.

[309] Verschueren, R., Bruyne, S. D., Zanon, M., Frasch, J. V., and Diehl, M. Towards time-optimal race car driving using nonlinear MPC in real-time. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2014), pp. 2505–2510.

[310] Verschueren, R., Zanon, M., Quirynen, R., and Diehl, M. A sparsity preserving convexification procedure for indefinite quadratic programs arising in direct optimal control. *SIAM Journal on Optimization (accepted, preprint available at Optimization Online, 2016-06-5512)* (2016).

[311] Verschueren, R., Zanon, M., Quirynen, R., and Diehl, M. Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm. In *Proceedings of the European Control Conference (ECC)* (2016).

[312] Villanueva, M. E., Quirynen, R., Diehl, M., Chachuat, B., and Houska, B. Robust MPC via min-max differential inequalities. *Automatica* (2016).

[313] Vukov, M. *Embedded Model Predictive Control and Moving Horizon Estimation for Mechatronics Applications*. PhD thesis, K.U. Leuven, 2015.

[314] Vukov, M., Domahidi, A., Ferreau, H. J., Morari, M., and Diehl, M. Auto-generated algorithms for nonlinear model predictive control on long and on short horizons. In *Proceedings of the IEEE Conference on Decision and Control (CDC)* (2013), pp. 5113–5118.

[315] Vukov, M., Gros, S., Horn, G., Frison, G., Geebelen, K., Jørgensen, J. B., Swevers, J., and Diehl, M. Real-time nonlinear MPC and MHE for a large-scale mechatronic application. *Control Engineering Practice 45* (2015), 64–78.

[316] Vukov, M., Loock, W. V., Houska, B., Ferreau, H. J., Swevers, J., and Diehl, M. Experimental validation of nonlinear MPC on an overhead crane using automatic code generation. In *Proceedings of the American Control Conference (ACC)* (2012), pp. 6264–6269.

[317] Wächter, A., and Biegler, L. IPOPT - an Interior Point OPTimizer. https://projects.coin-or.org/Ipopt, 2009.

[318] Wächter, A., and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*, 1 (2006), 25–57.

[319] Walther, A. *Program Reversal Schedules for Single- and Multi-processor Machines*. PhD thesis, TU Dresden, 2000.

[320] Walther, A. Automatic differentiation of explicit Runge-Kutta methods for optimal control. *Computational Optimization and Applications 36*, 1 (2006), 83–108.

[321] Walther, A., and Biegler, L. Numerical experiments with an inexact Jacobian trust-region algorithm. *Computational Optimization and Applications 48*, 2 (2011), 255–271.

[322] Walther, A., Vetukuri, S. R. R., and Biegler, L. T. A first-order convergence analysis of trust-region methods with inexact Jacobians and inequality constraints. *Optimization Methods and Software 27*, 2 (2012), 373–389.

[323] Wang, Y., and Boyd, S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology 18*, 2 (2010), 267–278.

[324] Wirsching, L. An SQP Algorithm with Inexact Derivatives for a Direct Multiple Shooting Method for Optimal Control Problems. Master's thesis, University of Heidelberg, 2006.

[325] Wirsching, L., Bock, H. G., and Diehl, M. Fast NMPC of a chain of masses connected by springs. In *Proceedings of the IEEE International Conference on Control Applications, Munich* (2006), pp. 591–596.

[326] Word, D. P., Kang, J., Akesson, J., and Laird, C. D. Efficient parallel solution of large-scale nonlinear dynamic optimization problems. *Computational Optimization and Applications 59*, 3 (2014), 667–688.

[327] Wright, S. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.

[328] Wynn, A., Vukov, M., and Diehl, M. Convergence guarantees for moving horizon estimation based on the real-time iteration scheme. *IEEE Transactions on Automatic Control 59*, 8 (2014), 2215–2221.

[329] Zanon, M. *Efficient Nonlinear Model Predictive Control Formulations for Economic Objectives and Applications in Aerospace and Autonomous Driving*. PhD thesis, KU Leuven, 2015.

[330] Zanon, M., Gros, S., and Diehl, M. Indefinite linear MPC and approximated economic MPC for nonlinear systems. *Journal of Process Control 24* (2014), 1273–1281.

[331] Zanon, M., Horn, G., Gros, S., and Diehl, M. Control of dual-airfoil airborne wind energy systems based on nonlinear MPC and MHE. In *Proceedings of the European Control Conference (ECC)* (2014), pp. 1801–1806.

[332] Zavala, V., and Anitescu, M. Real-Time Nonlinear Optimization as a Generalized Equation. *SIAM J. Control Optim. 48*, 8 (2010), 5444–5467.

[333] Zavala, V. M., and Biegler, L. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica 45* (2009), 86–93.

[334] Zavala, V. M., Laird, C. D., and Biegler, L. T. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chemical Engineering Science 63*, 19 (October 2008), 4834—-4845.

[335] Zometa, P., Kögel, M., and Findeisen, R. muAO-MPC: A free code generation tool for embedded real-time linear model predictive control. In *2013 American Control Conference* (June 2013), pp. 5320–5325.

[336] Zuo, W., and Lin, Z. A generalized accelerated proximal gradient approach for total-variation-based image restoration. *IEEE Transactions on Image Processing 20*, 10 (October 2011), 2748–2759.

Curriculum Vitae

Rien Quirynen was born in 1989 in Westmalle, Belgium. He studied Computer Science - Electrical Engineering during his Bachelor and completed his Master in Mathematical Engineering, both at the KU Leuven University in Belgium. During his Master studies, he gained experience in engineering research by doing a 2-month internship at LMS, Siemens in 2011 and by taking a summer job at IPCOS in 2012. For his Master project, he worked on the "Automatic code generation of Implicit Runge-Kutta integrators with continuous output for fast embedded optimization" under supervision of Prof. Diehl at KU Leuven. Since October 2012, he is pursuing his PhD at KU Leuven supported by a 4-year PhD scholarship from the Research Foundation - Flanders (FWO). As part of the YouReCa - Junior Mobility Programme, he has been a guest researcher at the University of Geneva (4 months) and at the Shanghai Jiao Tong University (3 months) during the year 2013-2014. Since May 2014, he has been affiliated with both KU Leuven and the University of Freiburg and therefore finished his project with a joint PhD between both universities.

List of Publications

Journal Articles (submitted)

1. Albin, T., Ritter, D., Quirynen, R., Diehl, M. (2016). In-Vehicle Realization of Nonlinear MPC for Gasoline Two-Stage Turbocharging Airpath Control. IEEE Transactions on Control Systems Technology.

Journal Articles (accepted)

2. Verschueren, R., Zanon, M., Quirynen, R., Diehl, M. (2016). A Sparsity Preserving Convexification Procedure for Indefinite Quadratic Programs Arising in Direct Optimal Control. SIAM Journal on Optimization.
3. Quirynen, R., Gros, S., Diehl, M. (2016). Inexact Newton-Type Optimization with Iterated Sensitivities. SIAM Journal on Optimization.
4. Quirynen, R., Gros, S., Houska, B., Diehl, M. (2016). Lifted Collocation Integrators for Direct Optimal Control in ACADO Toolkit. Mathematical Programming Computation.

Journal Articles (published)

5. Quirynen, R., Houska, B., Diehl, M. (2016). Efficient Symmetric Hessian Propagation for Direct Optimal Control. Journal of Process Control.
6. Gros, S., Zanon, M., Quirynen, R., Bemporad, A., Diehl, M. (2016). From Linear to Nonlinear MPC: bridging the gap via the Real-Time Iteration. International Journal of Control.
7. Villanueva, M. E., Quirynen, R., Diehl, M., Chachuat, B., Houska, B. (2016). Robust MPC via Min-Max Differential Inequalities. Automatica.

8. Quirynen, R., Vukov, M., Zanon, M., Diehl, M. (2014). Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. Optimal Control Applications and Methods.

9. Telen, D., Logist, F., Quirynen, R., Houska, B., Diehl, M., Van Impe, J. (2013). Optimal experiment design for nonlinear dynamic (bio)chemical systems using sequential semidefinite programming. AIChE (American Institute of Chemical Engineers) Journal.

Conference Proceedings

1. Kouzoupis, D., Quirynen, R., Girrbach, F., Diehl, M. (2016). An Efficient SQP Algorithm for Moving Horizon Estimation with Huber Penalties and Multi-Rate Measurements. IEEE Multi-Conference on Systems and Control. Buenos Aires, Argentina, September 2016.

2. Albin, T., Frank, F., Ritter, D., Abel, D., Quirynen, R., Diehl, M. (2016). Nonlinear MPC for Combustion Engine Control: A Parameter Study for Realizing Real-Time Feasibility. IEEE Multi-Conference on Systems and Control. Buenos Aires, Argentina, September 2016.

3. Verschueren, R., Duijkeren, N. V., Quirynen, R., Diehl, M. (2016). Exploiting Convexity in Direct Optimal Control: a Sequential Convex Quadratic Programming Method. Proc. 2016 IEEE 55th Annual Conference on Decision and Control. Las Vegas, USA, December 2016.

4. Kouzoupis, D., Quirynen, R., Garcia, J. L., Erhard, M., Diehl, M. (2016). A Quadratically Convergent Primal Decomposition Algorithm with Soft Coupling for Nonlinear Parameter Estimation. Proc. 2016 IEEE 55th Annual Conference on Decision and Control. Las Vegas, USA, December 2016.

5. Zanelli, A., Quirynen, R., Diehl, M. (2016). An Efficient Inexact NMPC Scheme with Stability and Feasibility Guarantees. Proc. of the 10th IFAC Symposium on Nonlinear Control Systems. NOLCOS 2016. Monterey, California, USA, August 2016.

6. Verschueren, R., Zanon, M., Quirynen, R., Diehl, M. (2016). Time-optimal Race Car Driving using an Online Exact Hessian based Nonlinear MPC Algorithm. Proc. of the European Control Conference. ECC 2016. Aalborg, Denmark, June 2016.

7. Quirynen, R., Houska, B., Diehl, M. (2016). Symmetric Hessian propagation for lifted collocation integrators in direct optimal control. Proc. of the American Control Conference. ACC 2016. Boston, MA, USA, July 2016.

8. Kouzoupis, D., Quirynen, R., Houska, B., Diehl, M. (2016). A Block Based ALADIN Scheme for Highly Parallelizable Direct Optimal Control. Proc. of the American Control Conference. ACC 2016. Boston, MA, USA, July 2016.

9. Quirynen, R., Gros, S., Diehl, M. (2015). Lifted implicit integrators for direct optimal control. Proc. 2015 IEEE 54th Annual Conference on Decision and Control. IEEE CDC 2015. Osaka, Japan, Dec. 2015.

10. Albin, T., Ritter, D., Abel, D., Quirynen, R., Diehl, M. (2015). Nonlinear MPC for a Two-Stage Turbocharged Gasoline Engine Airpath. Proc. 2015 IEEE 54th Annual Conference on Decision and Control. IEEE CDC 2015. Osaka, Japan, Dec. 2015.

11. Quirynen, R., Gros, S., Diehl, M. (2015). Inexact Newton based Lifted Implicit Integrators for fast Nonlinear MPC. Proc. of the 5th IFAC nonlinear model predictive control conference. NMPC 2015. Seville, Spain, September 2015.

12. Schmied, R., Waschl, H., Quirynen, R., Diehl, M., Del Re, L. (2015). Nonlinear MPC for Emission Efficient Cooperative Adaptive Cruise Control. Proc. of the 5th IFAC nonlinear model predictive control conference. NMPC 2015. Seville, Spain, September 2015.

13. Kouzoupis, D., Quirynen, R., Frasch, J., Diehl, M. (2015). Block Condensing for Fast Nonlinear MPC with the Dual Newton Strategy. Proc. of the 5th IFAC nonlinear model predictive control conference. NMPC 2015. Seville, Spain, September 2015.

14. Quirynen, R., Zanon, M., Kozma, A., Diehl, M. (2015). A Compression Algorithm for Real-Time Distributed Nonlinear MPC. Proc. of the European Control Conference. ECC 2015. Linz, Austria, July 2015.

15. Quirynen, R., Houska, B., Vallerio, M., Telen, D., Logist, F., Van Impe, J., Diehl, M. (2014). Symmetric Algorithmic Differentiation Based Exact Hessian SQP Method and Software for Economic MPC. Proc. 2014 IEEE 53rd Annual Conference on Decision and Control. IEEE CDC 2014. Los Angeles, USA, Dec. 2014.

16. Gros, S., Quirynen, R., Diehl, M. (2014). An Improved Real-time Economic NMPC Scheme for Wind Turbine Control Using Spline-Interpolated Aerodynamic Coefficients. Proc. 2014 IEEE 53rd Annual Conference on Decision and Control. IEEE CDC 2014. Los Angeles, USA, Dec. 2014.

17. Debrouwere, F., Vukov, M., Quirynen, R., Diehl, M., Swevers, J. (2014). Experimental validation of combined Nonlinear Optimal control and estimation of an overhead crane. Proc. of the 19th World Congress of the International Federation of Automatic Control. IFAC 2014. Cape Town, South Africa, Aug. 2014.

18. Quirynen, R., Gros, S., Diehl, M. (2013). Efficient NMPC for nonlinear models with linear subsystems. Proc. 2013 IEEE 52nd Annual Conference on Decision and Control. IEEE CDC 2013. Firenze, Italy, Dec. 2013.

19. Mehrkanoon, S., Quirynen, R., Diehl, M., Suykens, J.A.K. (2013). LS-SVM based initialization approach for parameter estimation of dynamical systems. Proc. 2013 International Conference on Mathematical Modeling in Physical Sciences. Prague, Czech Republic, Sept. 2013.

20. Quirynen, R., Gros, S., Diehl, M. (2013). Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. Proc. of the European Control Conference. ECC 2013. Zürich, Switzerland, Jul. 2013.

21. Gros, S., Quirynen, R., Diehl, M. (2012). Aircraft control based on fast nonlinear MPC and multiple-shooting. Proc. 2012 IEEE 51st Annual Conference on Decision and Control. IEEE CDC 2012. Maui, Hawai, Dec. 2012.

22. Quirynen, R., Vukov, M., Diehl, M. (2012). Auto generation of implicit integrators for embedded NMPC with microsecond sampling times. Proc. of the 4th IFAC nonlinear model predictive control conference. NMPC 2012. Noordwijkhout, Nederland, Aug. 2012.

Book Chapters

1. Quirynen, R., Vukov, M., Diehl, M. (2014). Multiple Shooting in a Microsecond. Contributions in Mathematical and Computational Sciences, Springer.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
Kasteelpark Arenberg 10 box 2446
B-3001 Leuven

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven

FACULTY OF MATHEMATICS AND PHYSICS
MATHEMATISCHES INSTITUT
Albert-Ludwigs-University Freiburg
D-79104 Freiburg im Breisgau